# CH569/CH565 Datasheet

V1.2

## Overview

The CH569 and CH565 both are based on the RISC-V3A core and they support the IMAC subset of RISC-V instruction set. On-chip 128-bit DMA is provided to meet the high bandwidth requirements for several high-speed peripherals and to implement high-speed transmission of large amounts of data. Peripherals include USB3.0 SuperSpeed and USB2.0 HighSpeed host and device controller and transceiver PHY, Gigabit Ethernet controller, dedicated high-speed SerDes controller and transceiver PHY, SD/EMMC interface controller, encryption/decryption module, high-speed parallel interface and digital video port (DVP), which can be widely used in streaming media, instant storage, super-high speed FIFO, communication expansion, security monitoring and other application scenarios.

## Features

- **Core**

- Support RV32IMAC instruction set combination, hardware multiplication and division
- Fast programmable interrupt controller + fast interrupt response
- Static branch prediction and conflict handler mechanism
- Low-power 2-stage pipeline
- Up to 120MHz system clock frequency

- **Memory**

- 448KB user application code memory CodeFlash
- 32KB user data memory DataFlash
- 24KB system boot program memory BootLoader
- 8KB system non-volatile configuration information memory InfoFlash
- 32/64/96KB configurable 128-bit SRAM (RAMX)
-16KB 32-bit SRAM (RAMS)

- **Power and low power**

- Support low power modes
- Support some GPIOs, USB and Ethernet signal wake-up

- **Timer and PWMX**

- 3 x 26-bit timers,
- Support timing, count, signal capture and PWM modulation output
- TMR1 and TMR2 support DMA
- 4 sets of extended PWM output, adjustable duty cycle

- **Universal asynchronous receiver transmitter (UART):**

- 4 UARTs, up to 7.5Mbps baud rate, compatible with 16C550
- Built-in FIFO, multiple trigger levels

- **Serial peripheral interface (SPI)**

- 2 SPI interfaces, support Master/Slave mode
- Built-in FIFO, support DMA

- **Active parallel port:**

  8-bit data, 15-bit address bus

- **General purpose I/O port**

- 49 ordinary IO ports, 8 of them support level/edge interrupts
- Support 3.3V or 2.5V, some pins have alternate and mapping functions

- **Others:** Watchdog, SysTick, debug interface, etc.

- **USB3.0 super-speed controller and transceiver (on-chip PHY)**

- Support USB3.0 Host/Device mode, OTG feature
- Support control, bulk, interrupt, isochronous transfer
- The host supports USB3.0 HUB
- Support U1/U2/U3 low power state

- **High-speed USB2.0 controller and transceiver (on-chip PHY)**

- Support USB2.0 Host/Device mode
- Support control, bulk, interrupt, isochronous transfer
- Support data receive/transmit buffer

- **Gigabit Ethernet controller (ETH)**

- Compliant with IEEE 802.3 protocol specifications
- Provide RGMII and RMII interfaces, connected to external PHY

- Support 10/100/1000Mbps transfer rate via PHY

- **EMMC controller**

- Conform to the UHS-ISDR50 mode in SD3.0 Specification and backward compatible
- Conform to EMMC card 4.4 and 4.5.1 Specifications, and compliant with EMMC card 5.0 Specification
- Support 1/4/8-line data communication, up to 96MHz communication clock

- **CH569 high speed parallel interface (HSPI)**

- Configurable 8/16/32-bit data
- Built-in FIFO, support DMA, dual buffer transceiver
- The fastest transfer rate is about 3.8Gbps (32-bit@120MHz)

- **CH565 digital video port (DVP)**

- Configurable 8/10/12 bit data
- Support YUV, RGB and JPEG compressed data

- **ECDC encryption module**

- Support AES/SM4 algorithm, with 8 combined encryption/decryption modes
- Support SRAM/EMMC/HSPI peripheral interface data encryption and decryption

- **Remote SerDes controller and transceiver (on-chip PHY)**

- 8b/10b encode/decode, 1.2Gbps high-speed differential signal communication
- 600Mbps transfer distance is up to 90m through a pair of differential network line

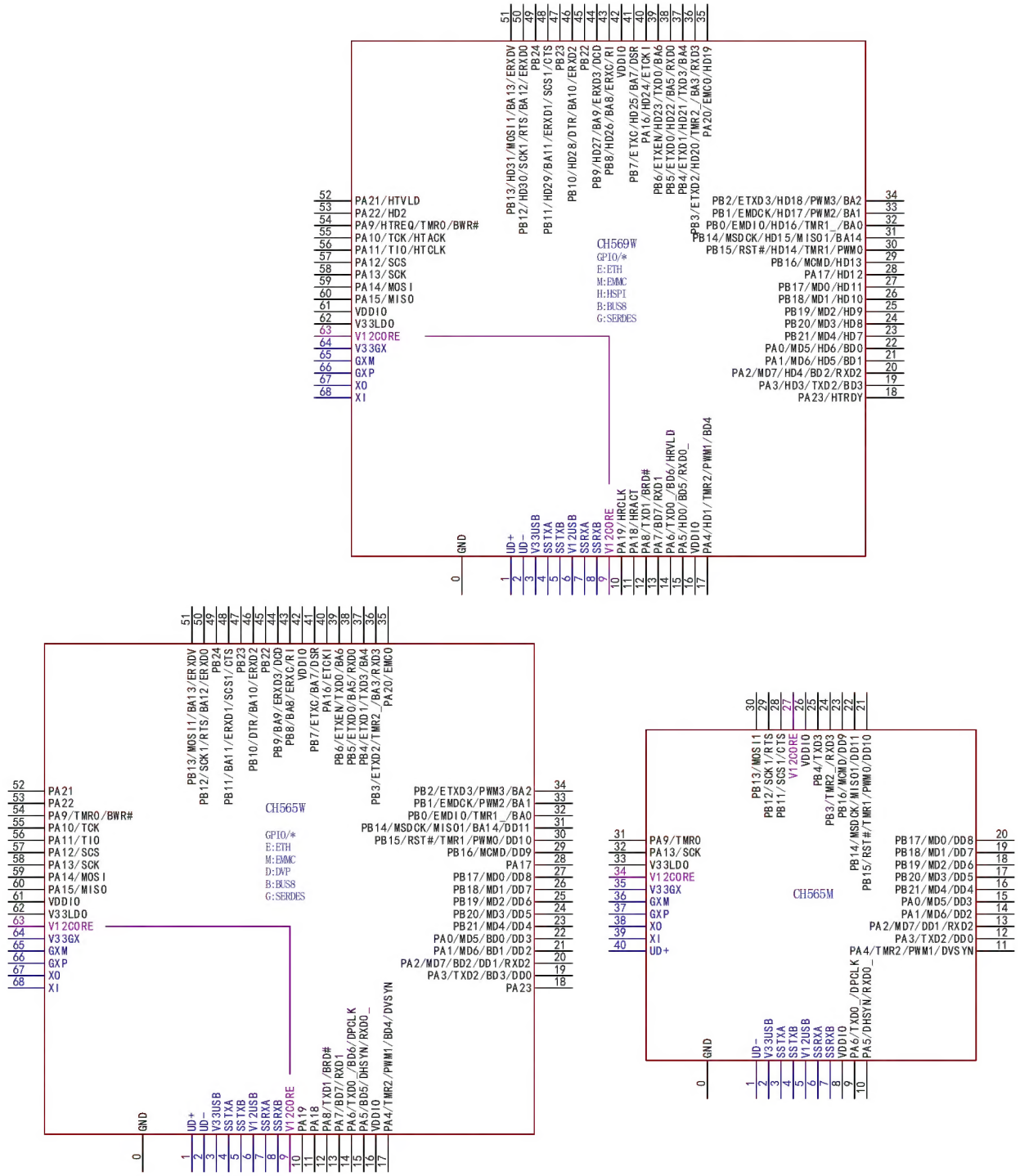
- **Unique ID:** Unique 64-bit ID

- **Package:** QFN68, QFN40

# Chapter 1 Pinouts and pin definitions

## 1.1 Pinouts

Figure 1-1 CH569 and CH565 pinouts




*Note 1: Pin 0# refers to the EPAD of QFN package.*

## 1.2 Comparison

Table 1-1 Comparison between CH569 and CH565

| Resource \ Part No. | | CH569W | CH565W | CH565M |
|---|---|---|---|---|
| Flash memory | CodeFlash | 448KB | 448KB | 448KB |
| | DataFlash | 32KB | | |
| | BootLoader | 24KB | | |
| SRAM (RAMX+RAMS) | | 32/64/96KB + 16KB | | |
| General purpose I/O | | 49 | 49 | 22 |
| Timer | General-purpose timer | 3 | 3 | 3 |
| | Watchdog (WDT) | √ | | |
| | SysTick | √ | | |
| PWMX+PWM | | 4 + 3 | 4 + 3 | 2 + 3 |
| UART | | 4 | 4 | 3 |
| SPI0 (master/slave) | | 2 | 2 | 1 |
| Active parallel port (BUS8) | | √ | √ | × |
| Gigabit Ethernet (ETH-GMAC) | | √ | √ | × |
| Digital video port (DVP) | | × | √ | √ |
| EMMC | | √ | √ | √ |
| AES/SM4 Encryption | | √ | √ | √ |
| High speed parallel interface (HSPI) | | √ | × | × |
| Super-speed USB3.0 | | √ | √ | √ |
| High-speed USB2.0 | | √ | √ | √ |
| High-speed SerDes | | √ | √ | √ |
| Debug interface | | √ | √ | × |

## 1.3 Pin definitions

### 1.3.1 CH569 pin definitions

Table 1-2 CH569 pin definitions

| Pin No. | Pin Name | Type | Main function (after reset)/ alternate and mapping | Description |
|---|---|---|---|---|
| 0 | GND | P | GND | Ground (EPAD) |
| 1 | UD+ | USB2.0 | DP | USB2.0 signal line D+ |
| 2 | UD- | USB2.0 | DN | USB2.0 signal line D- |
| 3 | V33USB | P | V33USB | 3.3V power input of USB PHY, connected to an external 0.1uF capacitor |
| 4 | SSTXA | USB3.0 | SSTXA | USB3.0 super-speed signal line TXA |
| 5 | SSTXB | USB3.0 | SSTXB | USB3.0 super-speed signal line TXB |
| 6 | V12USB | P | V12USB | 1.2V power input of USB PHY, connected to an external 0.1uF capacitor |
| 7 | SSRXA | USB3.0 | SSRXA | USB3.0 super-speed signal line RXA |

| | | | | |
|---|---|---|---|---|
| 8 | SSRXB | USB3.0 | SSRXB | USB3.0 super-speed signal line RXB |
| 9 | V12CORE | P | V12CORE | 1.2V core voltage input of the system, connected to the other V12CORE pin, an external 0.1uF capacitor is required |
| 10 | PA19 | I/O | PA19<br>/HRCLK | PA19: General purpose bidirectional digital I/O pin<br>HRCLK: Receive sampling clock input of high-speed parallel interface |
| 11 | PA18 | I/O | PA18<br>/HRACT | PA18: General purpose bidirectional digital I/O pin<br>HRACT: Transmit request signal input of high-speed parallel interface |
| 12 | PA8 | I/O | PA8<br>/TXD1/BRD# | PA8: General purpose bidirectional digital I/O pin<br>TXD1: TXD pin output of UART1 peripheral<br>BRD#: Read control signal output of active parallel interface, active low |
| 13 | PA7 | I/O | PA7<br>/BD7/RXD1 | PA78: General purpose bidirectional digital I/O pin<br>BD7: Data line 7 of active parallel interface<br>RXD1: RXD pin input of UART1 peripheral |
| 14 | PA6 | I/O | PA6<br>/TXD0_/BD6/HRVLD | PA6: General purpose bidirectional digital I/O pin<br>TXD0_: TXD function pin mapping of UART0 peripheral<br>BD6: Data line 6 of active parallel interface<br>HRVLD: Data transmit status pin input of high-speed parallel interface |
| 15 | PA5 | I/O | PA5<br>/HD0/BD5/RXD0_ | PA5: General purpose bidirectional digital I/O pin<br>HD0: High-speed parallel interface data line 0<br>BD5: Data line 5 of active parallel interface<br>RXD0_: RXD function pin mapping of UART0 peripheral |
| 16 | VDDIO | P | VDDIO | 3.3V or 2.5V input of I/O power, connected to an external 0.1uF capacitor |
| 17 | PA4 | I/O | PA4<br>/HD1/TMR2/PWM1/BD4 | PA4: General purpose bidirectional digital I/O pin<br>HD1: Data line 1 of high-speed parallel interface<br>TMR2: Input capture and PWM output pin of timer2 peripheral<br>PWM1: PWMX peripheral channel 1 output<br>BD4: Data line 4 of active parallel interface |
| 18 | PA23 | I/O | PA23<br>/HTRDY | PA23: General purpose bidirectional digital I/O pin<br>HTRDY: Detection receive status pin input of high-speed parallel interface |
| 19 | PA3 | I/O | PA3<br>/HD3/TXD2/BD3 | PA3: General purpose bidirectional digital I/O pin<br>HD3: Data line 3 of high-speed parallel interface<br>TXD2: TXD pin output of UART2 peripheral<br>BD3: Data line 3 of active parallel interface |
| 20 | PA2 | I/O | PA2<br>/MD7/HD4/BD2/RXD2 | PA2: General purpose bidirectional digital I/O pin<br>MD7: EMMC peripheral data line 7<br>HD4: Data line 4 of high-speed parallel interface<br>BD2: Data line 2 of active parallel interface |

| | | | | |
|---|---|---|---|---|
| | | | | RXD2: RXD pin input of UART2 peripheral |
| 21 | PA1 | I/O | PA1<br>/MD6/HD5/BD1 | PA1: General purpose bidirectional digital I/O pin<br>MD6: EMMC peripheral data line 6<br>HD5: Data line 5 of high-speed parallel interface<br>BD1: Data line 1 of active parallel interface |
| 22 | PA0 | I/O | PA0<br>/MD5/HD6/BD0 | PA0: General purpose bidirectional digital I/O pin<br>MD5: EMMC peripheral data line 5<br>HD6: Data line 6 of high-speed parallel interface<br>BD0: Data line 0 of active parallel interface |
| 23 | PB21 | I/O | PB21<br>/MD4/HD7 | PB21: General purpose bidirectional digital I/O pin<br>MD4: EMMC peripheral data line 5<br>HD7: Data line 7 of high-speed parallel interface |
| 24 | PB20 | I/O | PB20<br>/MD3/HD8 | PB20: General purpose bidirectional digital I/O pin<br>MD3: EMMC peripheral data line 3<br>HD8: Data line 8 of high-speed parallel interface |
| 25 | PB19 | I/O | PB19<br>/MD2/HD9 | PB19: General purpose bidirectional digital I/O pin<br>MD2: EMMC peripheral data line 2<br>HD9: Data line 9 of high-speed parallel interface |
| 26 | PB18 | I/O | PB18<br>/MD1/HD10 | PB18: General purpose bidirectional digital I/O pin<br>MD1: EMMC peripheral data line 1<br>HD10: Data line 10 of high-speed parallel interface |
| 27 | PB17 | I/O | PB17<br>/MD0/HD11 | PB17: General purpose bidirectional digital I/O pin<br>MD0: EMMC peripheral data line 0<br>HD11: Data line 11 of high-speed parallel interface |
| 28 | PA17 | I/O | PA17<br>/HD12 | PA17: General purpose bidirectional digital I/O pin<br>HD12: Data line 12 of high-speed parallel interface |
| 29 | PB16 | I/O | PB16<br>/MCMD/HD13 | PB16: General purpose bidirectional digital I/O pin<br>MCMD: Command signal line of EMMC peripheral<br>HD13: Data line 13 of high-speed parallel interface |
| 30 | PB15 | I/O | PB15<br>/RST#/HD14/TMR1/PWM0 | PB15: General purpose bidirectional digital I/O pin<br>RST#: External reset input pin, active low<br>HD14: Data line 14 of high-speed parallel interface<br>TMR1: Input capture and PWM output pin of timer1 peripheral<br>PWM0: PWMX peripheral channel 0 output |
| 31 | PB14 | I/O | PB14<br>/MSDCK/HD15/MISO1<br>/BA14 | PB14: General purpose bidirectional digital I/O pin<br>MCMD: Clock signal line output of EMMC peripheral<br>HD15: Data line 15 of high-speed parallel interface<br>MISO1: Host input slave output pin (MOSI) of SPI1 peripheral<br>BA14: Address line 14 of active parallel interface |
| 32 | PB0 | I/O | PB0<br>/EMDIO/HD16/TMR1_<br>/BA0 | PB0: General purpose bidirectional digital I/O pin<br>EMDIO: Ethernet controller SMI interface data pin<br>HD16: Data line 16 of high-speed parallel interface<br>TMR1_: Input capture and PWM output pin mapping of timer1 peripheral |

| | | | | |
|---|---|---|---|---|
| | | | | BA0: Address line 0 of active parallel interface |
| 33 | PB1 | I/O | PB1<br>/EMDCK/HD17/PWM2/BA1 | PB1: General purpose bidirectional digital I/O pin<br>EMDIO: Ethernet controller SMI interface clock output pin<br>HD17: Data line 17 of high-speed parallel interface<br>PWM2: PWMX peripheral channel 2 output<br>BA1: Address line 1 of active parallel interface |
| 34 | PB2 | I/O | PB2<br>/ETXD3/HD18/PWM3/BA2 | PB2: General purpose bidirectional digital I/O pin<br>EMDIO: Ethernet controller MII interface transmit data line 3<br>HD18: Data line 18 of high-speed parallel interface<br>PWM3: PWMX peripheral channel 3 output<br>BA2: Address line 2 of active parallel interface |
| 35 | PA20 | I/O | PA20<br>/EMCO/HD19 | PA20: General purpose bidirectional digital I/O pin<br>EMCO: Ethernet controller clock output pin<br>HD19: Data line 19 of high-speed parallel interface |
| 36 | PB3 | I/O | PB3<br>/ETXD2/HD20/TMR2_<br>/BA3/RXD3 | PB31: General purpose bidirectional digital I/O pin<br>ETXD2: Ethernet controller MII interface transmit data line 2<br>HD20: Data line 20 of high-speed parallel interface<br>TMR2_: Input capture and PWM output pin mapping of timer2 peripheral<br>BA3: Address line 3 of active parallel interface<br>RXD3: RXD pin input of UART3 peripheral |
| 37 | PB4 | I/O | PB4<br>/ETXD1/HD21/TXD3<br>/BA4 | PB4: General purpose bidirectional digital I/O pin<br>ETXD1: Ethernet controller MII interface transmit data line 1<br>HD21: Data line 21 of high-speed parallel interface<br>TXD3: TXD pin output of UART3 peripheral<br>BA4: Address line 4 of active parallel interface |
| 38 | PB5 | I/O | PB5<br>/ETXD0/HD22/BA5<br>/RXD0 | PB5: General purpose bidirectional digital I/O pin<br>EMDIO: Ethernet controller MII interface transmit data line 0<br>HD22: Data line 22 of high-speed parallel interface<br>BA5: Address line 5 of active parallel interface<br>RXD0: RXD pin input of UART0 peripheral |
| 39 | PB6 | I/O | PB6<br>/ETXEN/HD23/TXD0/BA6 | PB6: General purpose bidirectional digital I/O pin<br>ETXEN: Ethernet controller MII interface transmit data effective pin output<br>HD23: Data line 23 of high-speed parallel interface<br>TXD0: TXD pin output of UART0 peripheral<br>BA6: Address line 6 of active parallel interface |
| 40 | PA16 | I/O | PA16<br>/HD24/ETCKI | PA16: General purpose bidirectional digital I/O pin<br>HD24: Data line 24 of high-speed parallel interface<br>ETCKI: 125M clock input pin of Ethernet controller |
| 41 | PB7 | I/O | PB7<br>/ETXC/HD25/BA7<br>/DSR | PB7: General purpose bidirectional digital I/O pin<br>ETXC: Ethernet controller MII interface transmit clock pin output |

| | | | | |
|---|---|---|---|---|
| | | | | HD25: Data line 25 of high-speed parallel interface<br>BA7: Address line 7 of active parallel interface<br>DSR: DSR pin of UART0 peripheral MODEM signal |
| 42 | VDDIO | P | VDDIO | 3.3V or 2.5V input of I/O power, connected to an external 0.1uF capacitor |
| 43 | PB8 | I/O | PB8<br>/HD26/BA8/ERXC/RI | PB8: General purpose bidirectional digital I/O pin<br>HD26: Data line 26 of high-speed parallel interface<br>BA8: Address line 8 of active parallel interface<br>ETXC: Ethernet controller MII interface receive clock input pin<br>RI: RI pin of UART0 peripheral MODEM signal |
| 44 | PB9 | I/O | PB9<br>/HD27/BA9/ERXD3/DCD | PB9: General purpose bidirectional digital I/O pin<br>HD27: Data line 27 of high-speed parallel interface<br>BA9: Address line 9 of active parallel interface<br>ERXD3: Ethernet controller MII interface receive data line 3<br>DCD: DCD pin of UART0 peripheral MODEM signal |
| 45 | PB22 | I/O | PB22 | PB22: General purpose bidirectional digital I/O pin |
| 46 | PB10 | I/O | PB10<br>/HD28/DTR/BA10/ERXD2 | PB10: General purpose bidirectional digital I/O pin<br>HD28: Data line 28 of high-speed parallel interface<br>DTR: DTR pin of UART0 peripheral MODEM signal<br>BA10: Address line 10 of active parallel interface<br>ERXD2: Ethernet controller MII interface receiving data line 2 |
| 47 | PB23 | I/O | PB23 | PB23: General purpose bidirectional digital I/O pin |
| 48 | PB11 | I/O | PB11<br>/HD29/BA11/ERXD1<br>/SCS1/CTS | PB11: General purpose bidirectional digital I/O pin<br>HD29: Data line 29 of high-speed parallel interface<br>BA11: Address line 11 of active parallel interface<br>ERXD1: Ethernet controller MII interface receive data line 1<br>SCS1: Chip select signal pin (SCS) of SPI1 peripheral<br>CTS: CTS pin of UART0 peripheral MODEM signal |
| 49 | PB24 | I/O | PB24 | PB24: General purpose bidirectional digital I/O pin |
| 50 | PB12 | I/O | PB12<br>/HD30/SCK1/RTS<br>/BA12/ERXD0 | PB12: General purpose bidirectional digital I/O pin<br>HD30: Data line 30 of high-speed parallel interface<br>SCK1: Clock signal SCK pin of SPI1 peripheral<br>RTS: RTS pin of UART0 peripheral MODEM signal<br>BA12: Address line 12 of active parallel interface<br>ERXD0: Ethernet controller MII interface receive data line 0 |
| 51 | PB13 | I/O | PB13<br>/HD31/MOSI1/BA13<br>/ERXDV | PB13: General purpose bidirectional digital I/O pin<br>HD31: Data line 31 of high-speed parallel interface<br>MISI1: Master output slave input pin (MOSI) of SPI1 peripheral<br>BA13: Address line 13 of active parallel interface<br>ERXDV: Ethernet controller MII interface receive |

| | | | | |
|---|---|---|---|---|
| | | | | data effective pin input |
| 52 | PA21 | I/O | PA21<br>/HTVLD | PA21: General purpose bidirectional digital I/O pin<br>HTVLD: Data transmit status pin output of high-speed parallel interface |
| 53 | PA22 | I/O | PA22<br>/HD2 | PA22: General purpose bidirectional digital I/O pin<br>HD2: Data line 2 of high-speed parallel interface |
| 54 | PA9 | I/O | PA9<br>/HTREQ/TMR0/BWR# | PA9: General purpose bidirectional digital I/O pin<br>HTREQ: Transmit request signal output of high-speed parallel interface<br>TMR0: Input capture and PWM output pin of timer0 peripheral<br>BWR#: Write control signal output of active parallel interface, active low |
| 55 | PA10 | I/O | PA10<br>/TCK/HTACK | PA10: General purpose bidirectional digital I/O pin<br>TCK: Debug function clock input pin<br>HTACK: Receive status pin output of high-speed parallel interface |
| 56 | PA11 | I/O | PA11<br>/TIO/HTCLK | PA11: General purpose bidirectional digital I/O pin<br>TIO: Data input and output hardware of Debug function<br>HTCLK: Communication clock pin output of high-speed parallel interface |
| 57 | PA12 | I/O | PA12<br>/SCS | PA12: General purpose bidirectional digital I/O pin<br>SCS: Chip select signal pin (SCS) of SPI0 peripheral |
| 58 | PA13 | I/O | PA13<br>/SCK | PA13: General purpose bidirectional digital I/O pin<br>SCK: Clock signal SCK pin of SPI0 peripheral |
| 59 | PA14 | I/O | PA14<br>/MOSI | PA14: General purpose bidirectional digital I/O pin<br>MOSI: Master output slave input pin (MOSI) of SPI0 peripheral |
| 60 | PA15 | I/O | PA15<br>/MISO | PA15: General purpose bidirectional digital I/O pin<br>MISO1: Master input slave output pin (MISO) of SPI0 peripheral |
| 61 | VDDIO | P | VDDIO | 3.3V or 2.5V input of I/O power, connected to an external 0.1uF capacitor |
| 62 | V33LDO | P | V33LDO | 3.3V or 2.5V input of internal LDO power, connected to an external 0.1uF capacitor |
| 63 | V12CORE | P | V12CORE | 1.2V core voltage output of system, connected to an external 3.3uF capacitor |
| 64 | V33GX | P | V33GX | 3.3V input of SerDes PHY, connected to an external 0.1uF capacitor |
| 65 | GXM | SDP | GXM | SerDes differential signal line GXM |
| 66 | GXP | SDP | GXP | SerDes differential signal line GXP |
| 67 | XO | O | XO | Inverted output of external high-speed crystal oscillator |
| 68 | XI | I | XI | Input of external high-speed crystal oscillator |

### 1.3.2 CH565 pin definitions

Table 1-3 CH565 pin definitions

| Pin No. | | Pin Name | Type | Main Function (after reset)/ alternate and mapping | Description |
|---|---|---|---|---|---|
| CH569W | CH565M | | | | |
| 0 | 0 | GND | P | GND | Ground (EPAD) |
| 1 | 40 | UD+ | USB2.0 | DP | USB2.0 signal line D+ |
| 2 | 1 | UD- | USB2.0 | DN | USB2.0 signal line D- |
| 3 | 2 | V33USB | P | V33USB | 3.3V power input of USB PHY, connected to an external 0.1uF capacitor |
| 4 | 3 | SSTXA | USB3.0 | SSTXA | USB3.0 super-speed signal line TXA |
| 5 | 4 | SSTXB | USB3.0 | SSTXB | USB3.0 super-speed signal line TXB |
| 6 | 5 | V12USB | P | V12USB | 1.2V power input of USB PHY, connected to an external 0.1uF capacitor |
| 7 | 6 | SSRXA | USB3.0 | SSRXA | USB3.0 super-speed signal line RXA |
| 8 | 7 | SSRXB | USB3.0 | SSRXB | USB3.0 super-speed signal line RXB |
| 9 | 27 | V12CORE | P | V12CORE | 1.2V core voltage input of the system, connected to the other V12CORE pin with an external 0.1uF capacitor |
| 10 | - | PA19 | I/O | PA19 | PA19: General purpose bidirectional digital I/O pin |
| 11 | - | PA18 | I/O | PA18 | PA18: General purpose bidirectional digital I/O pin |
| 12 | - | PA8 | I/O | PA8 /TXD1/BRD# | PA8: General purpose bidirectional digital I/O pin<br>TXD1: TXD pin output of UART1 peripheral<br>BRD#: Read control signal output of active parallel interface, active low |
| 13 | - | PA7 | I/O | PA7 /BD7/RXD1 | PA78: General purpose bidirectional digital I/O pin<br>BD7: Data line 7 of active parallel interface<br>RXD1: RXD pin output of UART1 peripheral |
| 14 | 9 | PA6 | I/O | PA6 /TXD0_/BD6/DPCLK | PA6: General purpose bidirectional digital I/O pin<br>TXD0_: TXD function pin mapping of UART0 peripheral<br>BD6: Data line 6 of active parallel interface<br>DPCLK: Digital video port clock signal input |
| 15 | 10 | PA5 | I/O | PA5 /BD5/DHSYN/RXD0_ | PA5: General purpose bidirectional digital I/O pin<br>BD5: Data line 5 of active parallel interface<br>DHSYN: Sync signal input of digital video interface line<br>RXD0_: RXD function pin mapping of UART0 peripheral |
| 16 | 8 | VDDIO | P | VDDIO | 3.3V or 2.5V input of I/O power, connected with an external 0.1uF capacitor |
| 17 | 11 | PA4 | I/O | PA4 /TMR2/PWM1/BD4 /DVSYN | PA4: General purpose bidirectional digital I/O pin<br>TMR2: Input capture and PWM output pin of timer2 Peripheral |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | PWM1: PWMX peripheral channel 1 output<br>BD4: Data line 4 of active parallel interface<br>DVSYN: Sync signal input of digital video port frame |
| 18 | - | PA23 | I/O | PA23 | PA23: General purpose bidirectional digital I/O pin |
| 19 | 12 | PA3 | I/O | PA3<br>/TXD2/BD3/DD0 | PA3: General purpose bidirectional digital I/O pin<br>TXD2: TXD pin output of UART2 peripheral<br>BD3: Data line 3 of active parallel interface<br>DD0: Digital video port data line 0 |
| 20 | 13 | PA2 | I/O | PA2<br>/MD7/BD2/DD1<br>/RXD2 | PA2: General purpose bidirectional digital I/O pin<br>MD7: EMMC peripheral data line 7<br>BD2: Data line 2 of active parallel interface<br>DD1: Digital video port data line 1<br>RXD2: RXD pin input of UART2 |
| 21 | 14 | PA1 | I/O | PA1<br>/MD6/BD1/DD2 | PA1: General purpose bidirectional digital I/O pin<br>MD6: EMMC peripheral data line 6<br>BD1: Data line 1 of active parallel interface<br>DD2: Digital video port data line 2 |
| 22 | 15 | PA0 | I/O | PA0<br>/MD5/BD0/DD3 | PA0: General purpose bidirectional digital I/O pin<br>MD5: EMMC peripheral data line 5<br>BD0: Data line 0 of active parallel interface<br>DD3: Digital video port data line 3 |
| 23 | 16 | PB21 | I/O | PB21<br>/MD4/DD4 | PB21: General purpose bidirectional digital I/O pin<br>MD4: EMMC peripheral data line 4<br>DD4: Digital video port data line 4 |
| 24 | 17 | PB20 | I/O | PB20<br>/MD3/DD5 | PB20: General purpose bidirectional digital I/O pin<br>MD3: EMMC peripheral data line 3<br>DD5: Digital video port e data line 5 |
| 25 | 18 | PB19 | I/O | PB19<br>/MD2/DD6 | PB19: General purpose bidirectional digital I/O pin<br>MD2: EMMC peripheral data line 2<br>DD6: Digital video port data line 6 |
| 26 | 19 | PB18 | I/O | PB18<br>/MD1/DD7 | PB18: General purpose bidirectional digital I/O pin<br>MD1: EMMC peripheral data line 1<br>DD7: Digital video port data line 7 |
| 27 | 20 | PB17 | I/O | PB17<br>/MD0/DD8 | PB17: General purpose bidirectional digital I/O pin<br>MD0: EMMC peripheral data line 0<br>DD8: Digital video port data line 8 |
| 28 | - | PA17 | I/O | PA17 | PA17: General purpose bidirectional digital I/O pin |
| 29 | 23 | PB16 | I/O | PB16<br>/MCMD/DD9 | PB16: General purpose bidirectional digital I/O pin<br>MCMD: Command signal line of EMMC peripheral<br>DD9: Digital video port data line 9 |
| 30 | 21 | PB15 | I/O | PB15<br>/RST#/TMR1/PWM0<br>/DD10 | PB15: General purpose bidirectional digital I/O pin<br>RST# external reset input pin, active low<br>TMR1: Input capture and PWM output pin of timer1 peripheral<br>PWM0: PWMX peripheral channel 0 output<br>DD10: Digital video port data line 10 |
| 31 | 22 | PB14 | I/O | PB14 | PB14: General purpose bidirectional digital I/O pin |

| | | | | | |
|---|---|---|---|---|---|
| | | | | /MSDCK/MISO1<br>/BA14/DD11 | MCMD: Clock signal line output of EMMC peripheral<br>MISO1: Master input slave output pin (MOSI) of SPI1 peripheral<br>BA14: Address line 14 of active parallel interface<br>DD10: Digital video port data line 11 |
| 32 | - | PB0 | I/O | PB0<br>/EMDIO/TMR1_<br>/BA0 | PB0: General purpose bidirectional digital I/O pin<br>EMDIO: Ethernet controller SMI interface data pin<br>TMR1_: Input capture and PWM output pin mapping of timer1 peripheral<br>BA0: Address line 0 of active parallel interface |
| 33 | - | PB1 | I/O | PB1<br>/EMDCK/PWM2/BA1 | PB1: General purpose bidirectional digital I/O pin<br>EMDIO: Ethernet controller SMI interface clock output pin<br>PWM2: PWMX peripheral channel 2 output<br>BA1: Address line 1 of active parallel interface |
| 34 | - | PB2 | I/O | PB2<br>/ETXD3/PWM3/BA2 | PB2: General purpose bidirectional digital I/O pin<br>EMDIO: Ethernet controller MII interface transmit data line 3<br>PWM3: PWMX peripheral channel 3 output<br>BA2: Address line 2 of active parallel interface |
| 35 | - | PA20 | I/O | PA20<br>/EMCO | PA20: General purpose bidirectional digital I/O pin<br>EMCO: Ethernet controller clock output pin |
| 36 | 24 | PB3 | I/O | PB3<br>/ETXD2/TMR2_<br>/BA3/RXD3 | PB31: General purpose bidirectional digital I/O pin<br>ETXD2: Ethernet controller MII interface transmit data line 2<br>TMR2_: Input capture and PWM output pin mapping of timer 2 peripheral<br>BA3: Address line 3 of active parallel interface<br>RXD3: RXD pin input of UART3 peripheral |
| 37 | 25 | PB4 | I/O | PB4<br>/ETXD1/TXD3/BA4 | PB4: General purpose bidirectional digital I/O pin<br>ETXD1: Ethernet controller MII interface transmit data line 1<br>TXD3: TXD pin output of UART3<br>BA4: Address line 4 of active parallel interface |
| 38 | - | PB5 | I/O | PB5<br>/ETXD0/BA5/RXD0 | PB5: General purpose bidirectional digital I/O pin<br>EMDIO: Ethernet controller MII interface transmit data line 0<br>BA5: Address line 5 of active parallel interface<br>RXD0: RXD pin input of UART0 peripheral |
| 39 | - | PB6 | I/O | PB6<br>/ETXEN/TXD0/BA6 | PB6: General purpose bidirectional digital I/O pin<br>ETXEN: Ethernet controller MII interface transmit data effective pin output<br>TXD0: TXD pin output of UART0 peripheral<br>BA6: Address line 6 of active parallel interface |
| 40 | - | PA16 | I/O | PA16<br>/ETCKI | PA16: General purpose bidirectional digital I/O pin<br>ETCKI: 125M clock input pin of Ethernet controller |
| 41 | - | PB7 | I/O | PB7<br>/ETXC/BA7/DSR | PB7: General purpose bidirectional digital I/O pin<br>ETXC: Ethernet controller MII interface transmit |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | clock pin output<br>BA7: Address line 7 of active parallel interface<br>DSR: DSR pin of UART0 peripheral MODEM signal |
| 42 | 26 | VDDIO | P | VDDIO | 3.3V or 2.5V input of I/O power, connected to an external 0.1uF capacitor |
| 43 | - | PB8 | I/O | PB8<br>/BA8/ERXC/RI | PB8: General purpose bidirectional digital I/O pin<br>BA8: Address line 8 of active parallel interface<br>ETXC: Ethernet controller MII interface receive clock input pin<br>RI: RI pin of UART0 peripheral MODEM signal |
| 44 | - | PB9 | I/O | PB9<br>/BA9/ERXD3/DCD | PB9: General purpose bidirectional digital I/O pin<br>BA9: Address line 9 of active parallel interface<br>ERXD3: Ethernet controller MII interface receive data line 3<br>DCD: DCD pin of UART0 peripheral MODEM signal |
| 45 | - | PB22 | I/O | PB22 | PB22: General purpose bidirectional digital I/O pin |
| 46 | - | PB10 | I/O | PB10<br>/DTR/BA10/ERXD2 | PB10: General purpose bidirectional digital I/O pin<br>DTR: DTR pin of UART0 peripheral MODEM signal<br>BA10: Address line 10 of active parallel interface<br>ERXD2: Ethernet controller MII interface receive data line 2 |
| 47 | - | PB23 | I/O | PB23 | PB23: General purpose bidirectional digital I/O pin |
| 48 | 28 | PB11 | I/O | PB11<br>/BA11/ERXD1<br>/SCS1/CTS | PB11: General purpose bidirectional digital I/O pin<br>BA11: Address line 11 of active parallel interface<br>ERXD1: Ethernet controller MII interface receive data line 1<br>SCS1: Chip select signal pin (SCS) of SPI1 peripheral<br>CTS: CTS pin of UART0 peripheral MODEM signal |
| 49 | - | PB24 | I/O | PB24 | PB24: General purpose bidirectional digital I/O pin |
| 50 | 29 | PB12 | I/O | PB12<br>/SCK1/RTS<br>/BA12/ERXD0 | PB12: General purpose bidirectional digital I/O pin<br>SCK1: Clock signal SCK pin of SPI1 peripheral<br>RTS: RTS pin of UART0 peripheral MODEM signal<br>BA12: Address line 12 of active parallel interface<br>ERXD0: Ethernet controller MII interface receive data line 0 |
| 51 | 30 | PB13 | I/O | PB13<br>/MOSI1/BA13<br>/ERXDV | PB13: General purpose bidirectional digital I/O pin<br>MISI1: Master output slave input pin (MOSI) of SPI1 peripheral<br>BA13: Address line 13 of active parallel interface<br>ERXDV: Ethernet controller MII interface receive data effective pin input |
| 52 | - | PA21 | I/O | PA21 | PA21: General purpose bidirectional digital I/O pin |
| 53 | - | PA22 | I/O | PA22 | PA22: General purpose bidirectional digital I/O pin |
| 54 | 31 | PA9 | I/O | PA9<br>/TMR0/BWR# | PA9: General purpose bidirectional digital I/O pin<br>TMR0: Input capture and PWM output pin of timer0 peripheral<br>BWR#: Write control signal output of active parallel |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | interface, active low |
| 55 | - | PA10 | I/O | PA10<br>/TCK | PA10: General purpose bidirectional digital I/O pin<br>TCK: Clock input pin for Debug function |
| 56 | - | PA11 | I/O | PA11<br>/TIO | PA11: General purpose bidirectional digital I/O pin<br>TIO: Data input and output hardware of Debug function |
| 57 | - | PA12 | I/O | PA12<br>/SCS | PA12: General purpose bidirectional digital I/O pin<br>SCS: Chip select signal pin (SCS) of SPI0 peripheral |
| 58 | 32 | PA13 | I/O | PA13<br>/SCK | PA13: General purpose bidirectional digital I/O pin<br>SCK: Clock signal SCK pin of SPI0 peripheral |
| 59 | - | PA14 | I/O | PA14<br>/MOSI | PA14: General purpose bidirectional digital I/O pin<br>MOSI: Master output slave input pin (MOSI) of SPI0 peripheral |
| 60 | - | PA15 | I/O | PA15<br>/MISO | PA15: General purpose bidirectional digital I/O pin<br>MISO1: Master input slave output pin (MISO) of SPI0 peripheral |
| 61 | | VDDIO | P | VDDIO | 3.3V or 2.5V input of I/O power, connected to an external 0.1uF capacitor |
| 62 | 33 | V33LDO | P | V33LDO | 3.3V or 2.5V input of internal LDO power, connected to an external 0.1uF capacitor |
| 63 | 34 | V12CORE | P | V12CORE | 1.2V core voltage output of system, connected to an external 3.3uF capacitor |
| 64 | 35 | V33GX | P | V33GX | 3.3V input of SerDes PHY, connected to an external 0.1uF capacitor |
| 65 | 36 | GXM | SDP | GXM | SerDes differential signal line GXM |
| 66 | 37 | GXP | SDP | GXP | SerDes differential signal line GXP |
| 67 | 38 | XO | O | XO | Inverted output of external high-speed crystal oscillator |
| 68 | 39 | XI | I | XI | Input of external high-speed crystal oscillator |

*Note:*

*(1) I: Input; O: Output; I/O: Input and output.*

*(2) P: Power.*

*(3) USB: USB signal.*

*(4) SDP: SerDes PHY signal.*

# Chapter 2 System architecture and memory

## 2.1 System architecture

Figure 2-1 System architecture diagram

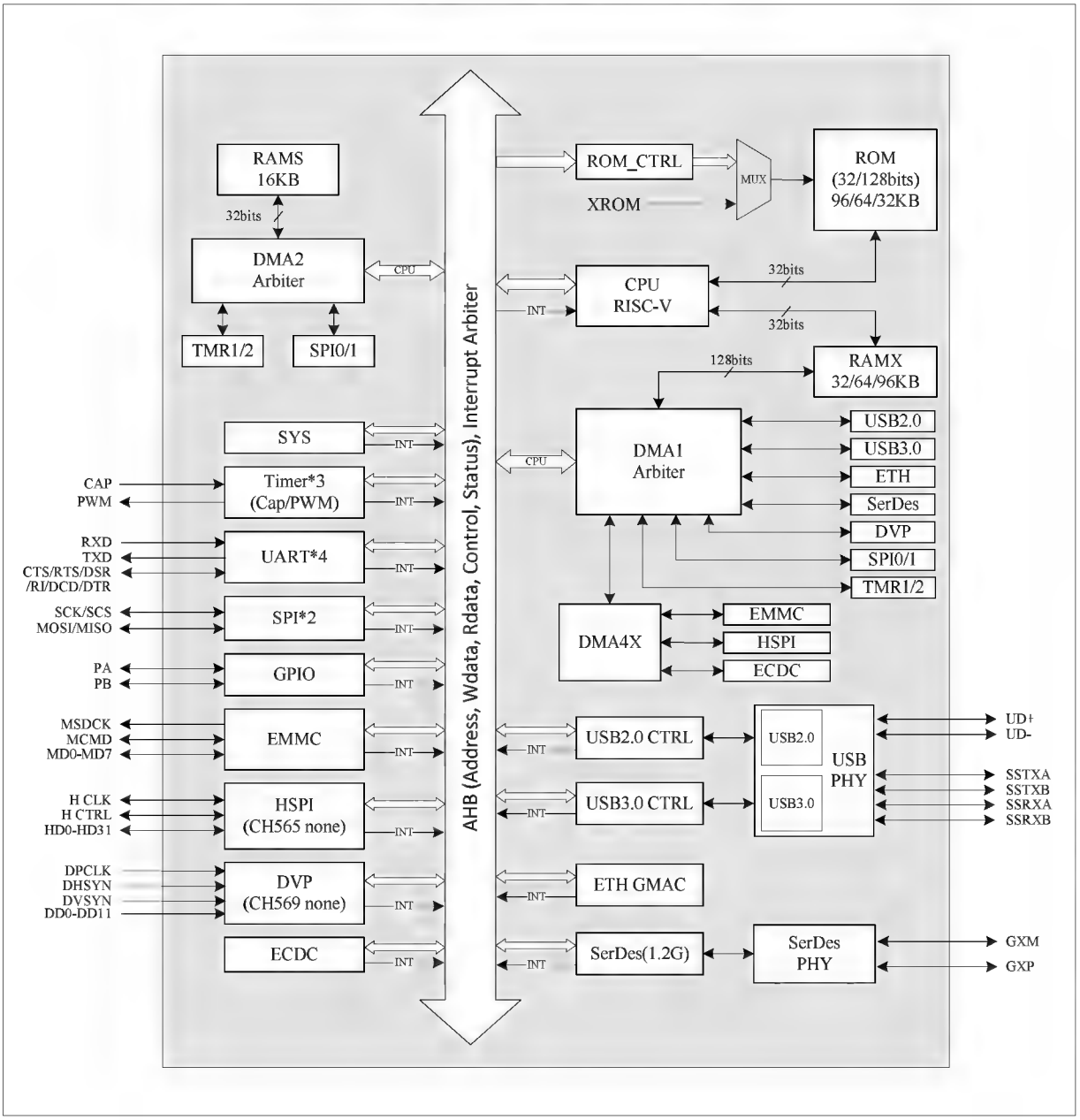



Figure 2-1 is the internal architecture diagram. The RISC-V core, DMA1/DMA2 arbitration controller, ROM control module, SRAM and various peripheral modules are mounted on the 32bit system bus (AHB). The core can access various peripherals and modules through the system bus, and receive external interrupt signals to trigger interrupt services. The system defaults to 32K bytes of ROM code at full speed without waiting, thus ensuring that the instruction addressing speed is consistent with the system clock frequency. The SRAM areas accessible to users are divided into the RAMS area and the RAMX area. Among them, the size of RAMX can be configured through the interface, and the extra area can be used for the SRAM mapping range of ROM code.

- RAMS: 16K bytes in total, 32-bit width of access SRAM, with the addressing range from 0x20000000

to 0x20003FFF

- RAMX: Configurable 32K/64K/96K byte size, 128-bit wide access high-speed SRAM, with the addressing range from 0x20020000 to 0x20037FFF
- DMA1: The access between RAMX and CPU, common peripherals and high-speed peripherals is established. High-speed peripherals: EMMC, HSPI, ECDC, USB2.0, USB3.0, ETH, SerDes and DVP.
- DMA2: The access between RAMS and CPU and common peripherals is established. Common peripherals include Timer1, Timer2, SPI0 and SPI1

*Note 1: CH569 is not equipped with DVP peripheral module. CH565 is not equipped with HSPI module. Other resources of them are the same.*
*Note 2: High-speed peripherals can only access the RAMX area. Pay attention to the DMA address range configured by high-speed peripheral registers for application code.*

## 2.2 Memory map

The addressing space of system bus mainly includes non-volatile memory (CODE), static memory (RAMS and RAMX), peripherals and external bus. Refer to Figure 2-2.

Figure 2-2 Memory map

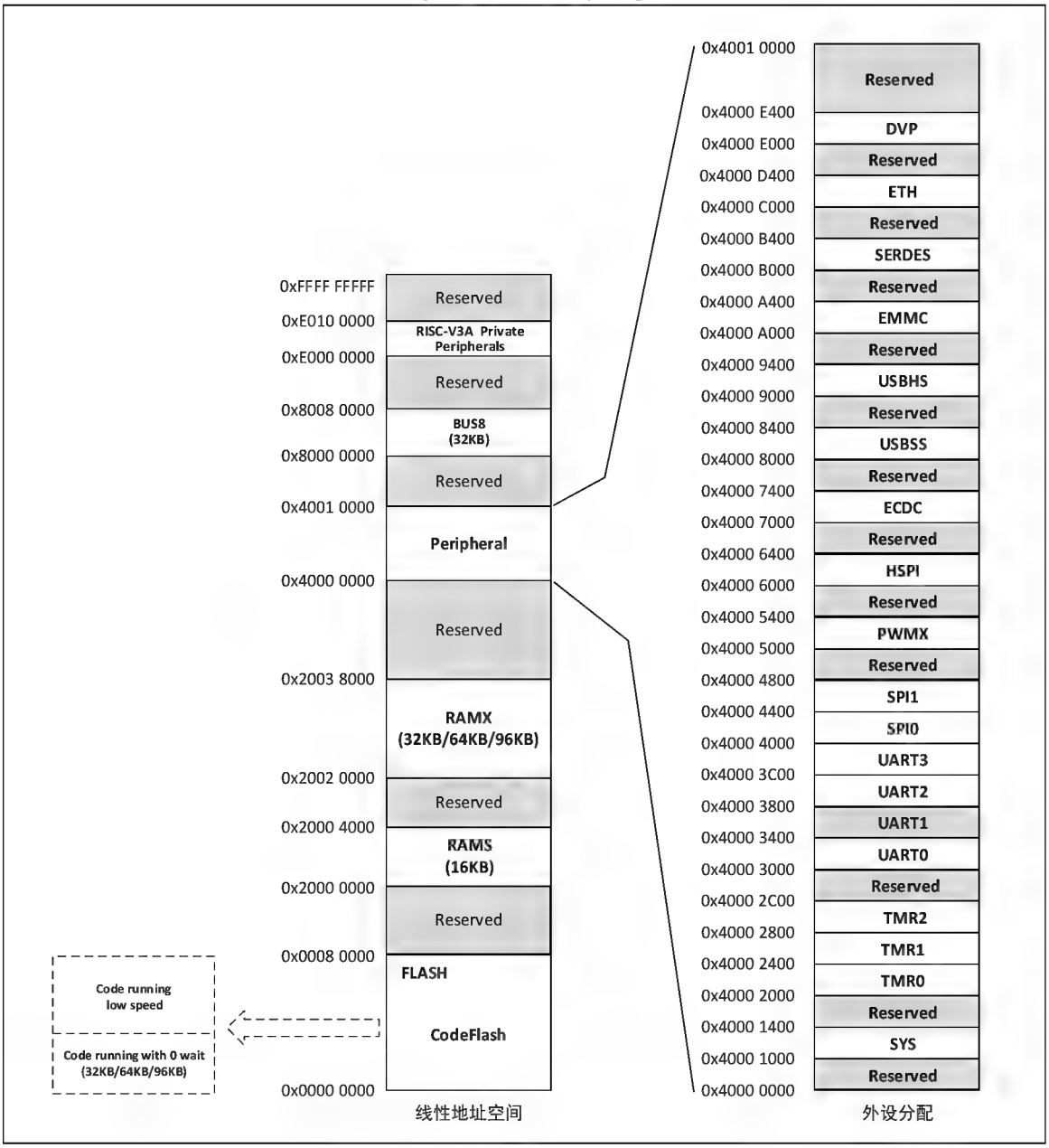



Table 2-1 Memory mapping area address

| Address range | Use | Description |
|---|---|---|
| 0x00000000-0x0007FFFF | On-chip non-volatile memory | Flash memory (512KB) |
| 0x00080000-0x1FFFFFFF | Reserved | - |
| 0x20000000-0X20003FFF | RAMS, usually used to store data | 16KB general SRAM |
| 0x20004000-0x2001FFFF | Reserved | - |
| 0x20020000-0x20037FFF | RAMX, used for DMA data access of high-speed peripherals | 32/64/96KB configurable SRAM |
| 0x20038000-0x3FFFFFFF | Reserved | - |
| 0x40000000-0x4000FFFF | Various functional modules or peripherals | Various peripheral address spaces |
| 0x40010000-0x7FFFFFFF | Reserved | - |
| 0x80000000-0x80007FFF | External bus | 32KB |
| 0x80008000-0xFFFFFFFF | Reserved | - |

### 2.2.1 On-chip non-volatile memory map

The non-volatile memory area mainly includes CodeFlash, DataFlash, BootLoader and InfoFlash. InfoFlash is used to store the manufacturer's data. Generally, the manufacturer writes the configuration information before the chip leaves the factory, as shown in Table 2-2.

Table 2-2 Description of non-volatile information

| Field | Name | Description | Default Value |
|---|---|---|---|
| [31:30] | USER_MEM | System RAMX/ROM capacity redefine configuration.<br>00: RAMX 32KB + ROM 96KB<br>01: RAMX 64KB + ROM 64KB<br>1x: RAMX 96KB + ROM 32KB<br>*Note: The above ROM space supports full-speed operation without waiting. If the size of code exceeds the above ROM space, the code of the excess part will run at a low speed of about 1/8.* | 10b |
| 29 | LOCKUP_RST_EN | Core LOCKUP reset system enable.<br>1: When the core generates LOCKUP, the system is reset;<br>0: When the core generates LOCKUP, the system is not reset. | 0 |
| 28 | RESERVED2 | Reserved. | 0 |
| [27:12] | RESERVED2 | Reserved. | FFFFh |
| [11:10] | RESERVED3 | Reserved. | 00b |
| [9:8] | RESERVED4 | Reserved. | 10b |
| 7 | CODE_READ_EN | External programmer read FLASH enable.<br>1: External programmer enabled to read FLASH (code is open);<br>0: External programmer disabled to read FLASH (code is protected). | 0 |
| 6 | BOOT_EN | Bootloader function enable.<br>1: BOOT program enabled;<br>0: BOOT program disabled. | 1 |
| 5 | DEBUG_EN | Debug interface enable.<br>1: Debug interface enabled;<br>0: Debug interface disabled.<br>*Note: When debug interface is enabled, external reset must be disabled. Otherwise, the debug function cannot be used normally.* | 0 |
| 4 | RESET_EN | External reset enable.<br>1: External reset input from PB15 enabled, and PB15 is used as system external reset pin;<br>0: External reset input from PB15 disabled, and PB15 is used as a normal GPIO. | 0 |
| [3:0] | RESERVED5 | Reserved. | 0101b |

### 2.2.2 On-chip DATA area map

Table 2-3 On-chip static memory allocation

| Address range | Description | Size |
|---|---|---|
| 0x20000000-0x20003FFF | RAMS area, CPU access variables and low-speed peripherals access DMA data. | 16KB |
| 0x20020000-0x20027FFF | When USER_MEM=00b, the RAMX area addressed by the high-speed peripheral DMA | 32KB |
| 0x20020000-0x2002FFFF | When USER_MEM=01b, the RAMX area addressed by the high-speed peripheral DMA | 64KB |
| 0x20020000-0x20037FFF | When USER_MEM=1xb, the RAMX area addressed by the high-speed peripheral DMA | 96KB |

The system contains abundant SRAM resources for the CPU to access variables and stack data, and it meets the DMA data access requirements of various peripherals. The internal SRAM resources are mainly divided into 2 types: one type is mainly used for CPU access variables and low-speed peripheral DMA data access RAMS; the other type is mainly used for high-speed peripheral DMA data access RAMX (SRAM1+SRAM2). CPU, SPI and TMR can access all the resources of RAMS and RAMX by correctly setting the DMA address, but all other high-speed peripheral DMA can only use RAMX resources, and no longer compete with CPU memory access behavior. Pay attention to the address allocation of data access when different peripherals are used for application code.

RAMS area access bit width is 32bits, and the start address is 0x20000000, totaling 16KB. RAMX area access bit width is 128bits, and the start address is 0x20020000, and the size depends on the setting of user configuration word USER_MEM, and 3 flexible capacity choices of 32KB/64KB/96KB are supported.

### 2.2.3 Unique ID and Flash operation

Each chip has a unique ID (chip identification number) when it is delivered from the factory. This ID data and its checksum has 8 bytes in total, stored in the read-only area of chip. Please refer to routines for specific operations.

Library functions can be provided for the operation of DataFlash and CodeFlash. Please refer to routines.

### 2.2.4 Peripheral address assignment

The system is equipped with 18 peripherals and 1 external bus interface. Each peripheral occupies a certain address space, and the actual access address of peripheral register is: base address + offset address. In the following chapters, the address of the register is described separately by base address and offset address.

The following table shows the explanation of "Access" in the register description in this datasheet:

| Attribute | Description |
|---|---|
| RF | Software can only read this bit. The read value is fixed, which is not affected by reset. |
| RO | Software can only read these bits. Changed by hardware. |
| RZ | Software can only read this bit. Reading this bit automatically clears it to 0. |
| WO | Software can only write to these bits (unreadable, uncertain read value) |

| | |
|---|---|
| WA | Software can only write to these bits. Write in safe mode. |
| WZ | Software can only write to these bits. Automatically cleared after write operation. |
| RW | Software can read and write to these bits. |
| RWA | Software can read this bit as well as write in safe mode. |
| RW1 | Software can read this bit. Valid when writing 1. Invalid when writing 0. |
| RW0 | Software can read this bit. Valid when writing 0. Invalid when writing 1. |
| RW1T | Software can read this bit. Invalid when writing 0, toggle when writing 1. |

# Chapter 3 System control

## 3.1 Power control

The power control of the system is designed to separately supply power to different areas. Therefore, multiple 3.3V voltage sources need to be provided externally, to ensure the normal operation of the core, GPIOs, internal LDO, built-in USB PHY and built-in high-speed SerDes PHY.

- VDDIO: System IO port power input. Several groups of VDDIO supply power to all IO ports.
- V33LDO: Power input of system internal LDO voltage regulator module.
- V12CORE: 1.2V voltage output and input of the core.
- V33GX: SerDes module transceiver power.
- V33USB, V12USB: USB3.0 and USB2.0 module transceiver power.

After system reset, the microcontroller runs normally. When the CPU does not need to continue to run, or when some functional modules do not need to be used, either the clock or independent power of these modules can be turned off, to reduce power consumption.

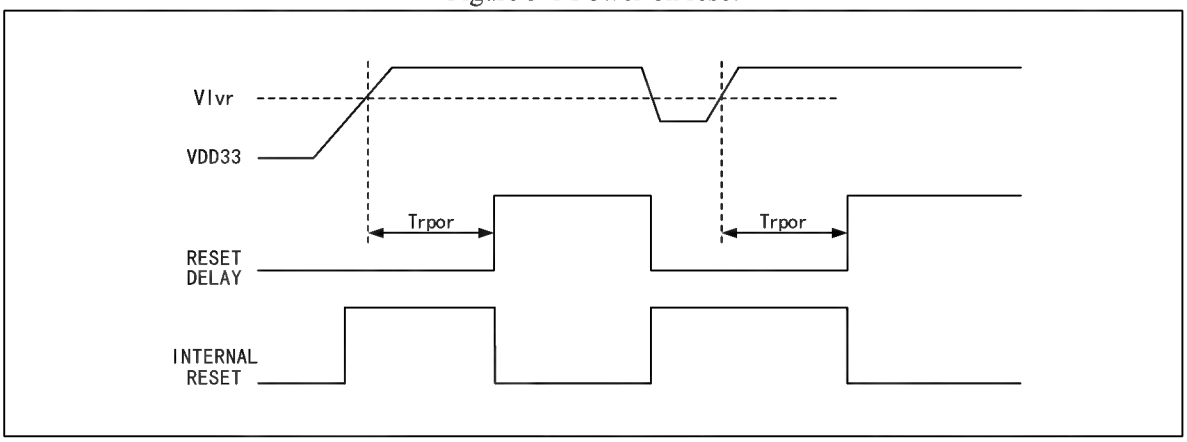
## 3.2 Reset control

The system provides 4 types of reset: power-on reset, manual reset, software reset and watchdog reset. Only power-on reset will cause the system to reload configuration information and reload the program code into SRAM. The R8_RST_BOOT_STAT register records the reset source of the last reset. The R8_GLOB_RESET_KEEP register when power on reset occurs, and it is not affected by other types of reset.

### 3.2.1 Power-on reset

When the power is turned on, the internal POR module generates power-on reset timing and it delays to wait for the power to stabilize. During operation, when the power voltage is lower than $V_{lvr}$, the internal LVR module generates a low voltage reset until the voltage rises, and it delays to wait for the power to stabilize. Figure 3-1 shows the power-on reset process and the low-voltage reset process.

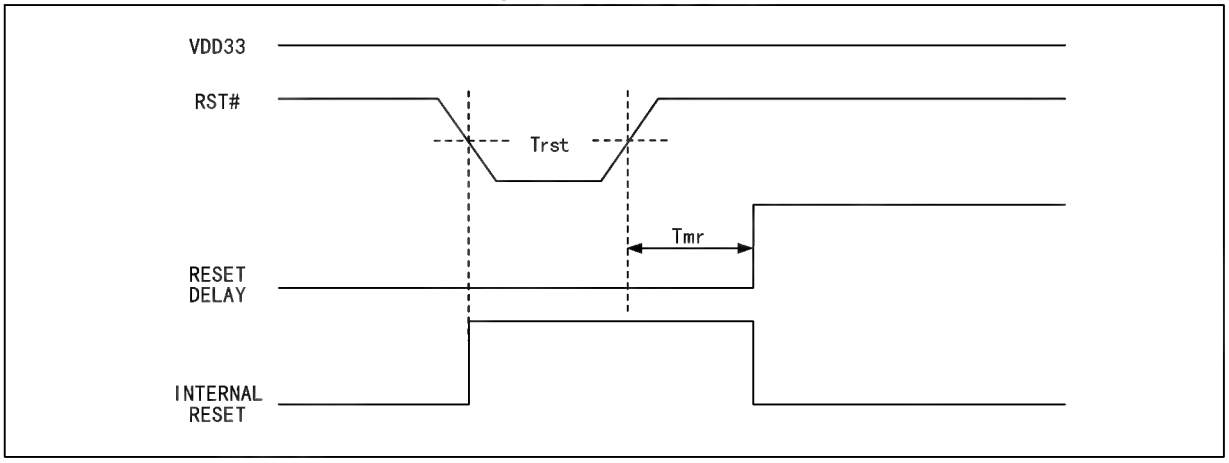
Figure 3-1 Power-on reset




### 3.2.2 Manual reset

The external manual reset is generated by the low level applied to the RST# pin from the external. When the duration of external low-level reset signal is longer than the minimum reset pulse width (Trst), the system reset is triggered, and the reset process is completed after $T_{mr}$.

Figure 3-2 External reset




### 3.2.3 Software reset

The system provides 2 types of internal software reset, so that no external intervention is required to perform software reset in some specific situations.

1) Set the RB_SOFTWARE_RESET bit in the reset configuration register (R8_RST_WDOG_CTRL) to 1, and this bit is automatically cleared.

2) Reset the system by configuring the SYSRESET bit in the core interrupt configuration register (PFIC_CFGR) to 1, and refer to the corresponding chapter for details.

### 3.2.4 Watchdog reset

The watchdog feature is based on an 8-bit upcounter, and the counting clock time base is Fsys/524288. When the watchdog timeout reset function is enabled (RB_WDOG_RST_EN=1), system reset will be triggered once this counter overflows.

### 3.2.5 Reset feature

Please refer to Chapter 19 for reset parameters.

## 3.3. Clock control

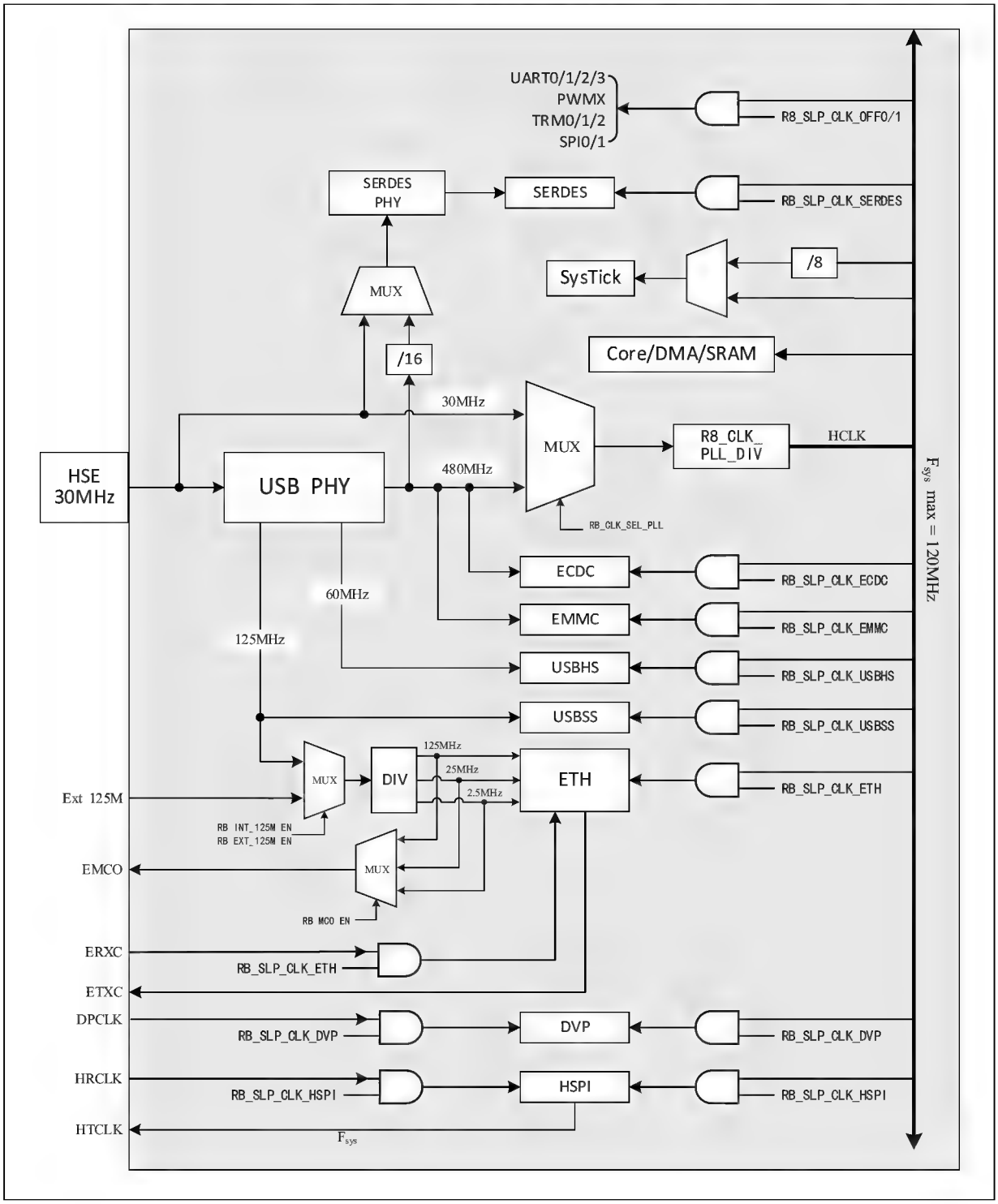
When the system is working, an external 30MHz high-speed clock signal (HSE) is needed, which can be provided from either external crystal/ceramic resonator or external high-speed clock input.

As shown in Figure 3-3, 30MHz clock is sent to the USB PHY module inside the system. This module will generate several clock frequencies, including 125MHz clock for USB3.0 controller and internal clock source of Ethernet module, 60MHz clock for USB2.0 controller and 480MHz clock as PLL clock frequency ($F_{pll}$).

- The system bus clock (HCLK) comes from $F_{pll}$ frequency division or HSE frequency division, which is determined by the configurations of the R8_CLK_CFG_CTRL register and the R8_CLK_PLL_DIV register. Its frequency is $F_{sys}$ ($T_{sys}$). HCLK clock acts on the access of all registers and SRAM of the system, the digital logic operation of each peripheral module, DMA, and the operation of each unit of the core.
- The high-speed parallel interface module (HSPI) implements data transmission via HCLK clock, and the output HTCLK frequency is equal to $F_{sys}$.

- The reference clock $F_{SDSP}$ of high-speed SerDes PHY comes from HSE or $F_{pll}$ divided by 16. $F_{SDSP}$ is multiplied to generate the clock $F_{SDS}$ required by SerDes controller for SerDes to receive/transmit data.
- The 3 clocks of 125MHz/25MHz/2.5MHz required by the Ethernet controller come from either the internal 125MHz clock or the 125MHz clock input by the external ETCKI pin, which is determined by the configurations of the R8_CLK_MOD_AUX register. In this case, software can configure it to output 3 clocks from the EMCO pin for connecting to an external Ethernet PHY.
- Each peripheral module clock can be independently turned on and turned off. In order to reduce the power consumption of chip, function module clocks that are not used can be turned off. Refer to the description of the R8_SLP_CLK_OFF0 register and the R8_SLP_CLK_OFF1 register for details.

Figure 3-3 Clock tree



## 3.4 System clock configuration

The system uses (HSE/2) as the system clock by default, which is 15MHz. The application code can obtain $F_{sys}$ which ranges from 2MHz to 120MHz by configuring the R8_CLK_CFG_CTRL register and the R8_CLK_PLL_DIV register.

When RB_CLK_SEL_PLL=1, the system clock is divided by 480MHz, and the division factor is R8_CLK_PLL_DIV.
When RB_CLK_SEL_PLL=0, the system clock is divided by 30MHz, and the division factor is R8_CLK_PLL_DIV.

It is required to note the followings in the process of configuring the system clock:

1) When configuring R8_CLK_CFG_CTRL, bits [7:6] must be 01b. Otherwise, the configuration of this register has no effect.
2) When configuring R8_CLK_PLL_DIV, bits [7:6] must be 10b. Otherwise, the configuration of this register has no effect.
3) Before the system clock source is switched from 30MHz to 480MHz, the division factor must be set correctly. If the clock source is switched directly, the high-frequency clock may cause unexpected abnormality to the system.

## 3.5 Low power modes and wake-up

At the end of system reset, the microcontroller works normally. When MCU does not need to run, it can enter low power mode to save power consumption. The system provides 3 low power modes: Idle mode, Halt mode and Sleep mode.

- Idle mode: All peripherals keep running normally, while the core stops running, and the clock system and peripheral modules run normally. When a wake-up event is detected, it can wake up immediately within the shortest time. If there are unnecessary peripherals, the corresponding peripheral clock control bit can be disabled to reduce power consumption.
- Halt mode: On the basis of idle mode, PLL stops working, the clock system no longer runs and the related peripherals are stopped. Most of power consumption can be reduced. After detecting a wake-up event, first wake up the system clock, then wake up the core, and the system will restart.
- Sleep mode: On the basis of the halt mode, the crystal oscillation is stopped to reach the lowest power consumption. After wakeup in this mode, it can work normally after the crystal oscillator is stabilized, so a longer wake-up time is required.

In low power mode, the system supports 4 wakeup conditions including part GPIO port wakeup, high-speed USB2.0 wakeup, super-speed USB3.0 wakeup, and Ethernet wake-up. Refer to the R8_SLP_WAKE_CTRL register. The following table describes the features and wake-up methods of 3 low power modes in detail:

Table 3-1 Low power modes

| Mode | Feature | Entry conditions | Wake-up event |
|---|---|---|---|
| Idle | Peripherals are running normally, the core stops running (the core clock stops), and the clock system is running normally, but the clock of each peripheral can be turned off by | Set the SleepDeep domain in the PFIC_SCTLR register to 0. Execute __WFI() or __WFE() after setting the wake-up conditions. | All wakeup sources |

| | | | |
|---|---|---|---|
| | the peripheral clock control bit. | | |
| Halt | The core stops running, the PLL stops working, the clock system no longer runs, and all related peripherals stop. | Set the SleepDeep domain in the PFIC_SCTLR register to 1. Set RB_SLP_USBHS_PWRDN to 1. Execute __WFI() or __WFE() after setting the wake-up conditions. | All wakeup sources |
| Sleep | The core stops running, the PLL stops working, and the external crystal oscillator stops oscillating. Reach the lowest power consumption. | Set the SleepDeep domain in the PFIC_SCTLR register to 1. Set RB_SLP_USBHS_PWRDN to 1. USB3.0 controller enters low power mode (refer to the routines). Execute __WFI() or __WFE() after setting the wake-up conditions. | All wakeup sources |

## 3.6 Register description

System control register base address: 0x40001000

Table 3-2 System control registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_SAFE_ACCESS_SIG | 0x00 | Safe access flag register | 0x00 |
| R8_CHIP_ID | 0x01 | Chip ID register | - |
| R8_SAFE_ACCESS_ID | 0x02 | Safe access ID register | 0x02 |
| R8_WDOG_COUNT | 0x03 | Watchdog counter register | 0x00 |
| R8_GLOB_ROM_CFG | 0x04 | ROM configuration register | 0x80 |
| R8_RST_BOOT_STAT | 0x05 | BOOT status register | 0xXX |
| R8_RST_WDOG_CTRL | 0x06 | Reset control register | 0x00 |
| R8_GLOB_RESET_KEEP | 0x07 | Reset keep register | 0x00 |
| R8_CLK_PLL_DIV | 0x08 | PLL output clock divider register | 0x42 |
| R8_CLK_CFG_CTRL | 0x0A | Clock configuration register | 0x80 |
| R8_CLK_MOD_AUX | 0x0B | Clock auxiliary register | 0x00 |
| R8_SLP_CLK_OFF0 | 0x0C | Sleep control register 0 | 0x00 |
| R8_SLP_CLK_OFF1 | 0x0D | Sleep control register 1 | 0x00 |
| R8_SLP_WAKE_CTRL | 0x0E | Wakeup control register | 0x00 |
| R8_SLP_POWER_CTRL | 0x0F | Low power control register | 0x00 |
| R16_SERD_ANA_CFG1 | 0x20 | SerDes PHY analog parameter configuration register 1 | 0x005A |
| R32_SERD_ANA_CFG2 | 0x24 | SerDes PHY analog parameter configuration register 2 | 0x00423015 |

Safe access flag register (R8_SAFE_ACCESS_SIG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_SAFE_ACCESS_SIG | WO | Safe access flag register. Some registers (access attribute is RWA) are protection registers, and write operation can be | 0 |

| | | | | |
|---|---|---|---|---|
| | | | conducted only after entering the safe access mode. Write 0x57 first and then 0xA8 to this register, to enter the safe access mode. The time is limited to about 16 system clock cycles (Tsys), and it will be automatically protected beyond the time limit. You can write any other value to exit the safe access mode directly and return to the protected state. | |
| [6:4] | RB_SAFE_ACC_TIMER | RO | Current safe access time count. | 0 |
| [1:0] | RB_SAFE_ACC_MODE | RO | Current safe access mode status:<br>11: Safe mode, registers with RWA attribute can be accessed;<br>Others: Non-safe mode, write operations to protect registers disabled. | 0 |

Chip ID register (R8_CHIP_ID)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_CHIP_ID | RF | Used to identify the chip. | - |

Security access ID register (R8_SAFE_ACCESS_ID)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_SAFE_ACCESS_ID | RF | Fixed to 02h. | 02h |

Watchdog counter register (R8_WDOG_COUNT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_WDOG_COUNT | RW | Watchdog counter whose initial value can be preset will keep incrementing automatically, and it can cycle from 0xFF to 0x00 and then continue.<br>Count time base = Fsys/524288. | 0 |

ROM configuration register (R8_GLOB_ROM_CFG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RWA | Reserved. Must write 10b to [7:6] bits. Read as 0. | 0 |
| 4 | RB_ROM_CODE_OFS | RWA | Configure the start offset address of user code in FlashROM. This value can only be cleared when power-on reset and is not affected by other resets.<br>1: 0x04000; 0: 0x00000. | 0 |
| 3 | RB_ROM_CODE_WE | RWA | Flash ROM code and data area erase/write enable.<br>1: Program/erase; 0: Write protection. | 0 |
| 2 | RB_ROM_DATA_WE | RWA | Flash ROM data area erase/write enable.<br>1: Program/erase; 0: Write protection. | 0 |
| 1 | RB_CODE_RAM_WE | RWA | Code SRAM area write enable bit.<br>1: Write enabled; 0: Write protection. | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | RB_ROM_EXT_RE | RO | External programmer read Flash ROM enable.<br>1: Read enabled; 0: Read protection. | 0 |

BOOT status register (R8_RST_BOOT_STAT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:6] | Reserved | RO | Reserved. | 11b |
| 5 | RB_BOOT_LOADER | RO | Boot loader status<br>1: BootLoader;<br>0: User code. | 0 |
| 4 | RB_CFG_DEBUG_EN | RO | Debug enable<br>1: Enabled; 0: Disabled. | 0 |
| 3 | RB_CFG_BOOT_EN | RO | Bootloader enable<br>1: Enabled; 0: Disabled. | 1 |
| 2 | RB_CFG_RESET_EN | RO | External reset pin (RST#) enable<br>1: External input low-level signal reset;<br>0: Disabled. | 0 |
| [1:0] | RB_RESET_FLAG | RO | Last reset status<br>00: Software reset. Source: Internal software request reset and RB_BOOT_LOADER=0.<br>01: Power-on reset. Source: Chip operating voltage is lower than the threshold.<br>10: Watchdog reset. Source: Watchdog counter overflow.<br>11: Manual reset. Source: RST# pin input low level. | xx |

Reset control register (R8_RST_WDOG_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:4] | Reserved | RO | Reserved. Must write 01b to [7:6] bits. | 0 |
| 3 | RB_WDOG_INT_FLAG | RW1 | Watchdog counter overflow flag.<br>1: Watchdog count overflows, that is, R8_WDOG_COUNT is detected to progressively increase from 0xFF to 0x00;<br>0: Watchdog count not overflow.<br>Clear by writing 1. Automatically cleared when the watchdog counter value is reloaded. | 0 |
| 2 | RB_WDOG_INT_EN | RWA | Watchdog counter overflow interrupt enable.<br>1: Watchdog counter overflow interrupt enabled;<br>0: Watchdog counter overflow interrupt disabled. | 0 |
| 1 | RB_WDOG_RST_EN | RWA | Watchdog counter overflow reset enable.<br>1: Reset when the counter overflows;<br>0: Not reset when the counter overflows. | 0 |
| 0 | RB_SOFTWARE_RESET | WA/ WZ | System software reset. Cleared automatically:<br>1: System reset; 0: No action. | 0 |

Reset keep register (R8_GLOB_RESET_KEEP)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_GLOB_RESET_KEEP | RW | Reset keep register. The value of this register can be reset only by power-on reset. This register is not affected by manual reset, software reset or watchdog reset. | 0 |

PLL output clock divider register (R8_CLK_PLL_DIV)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_CLK_PLL_DIV | RWA | The lower 4 bits are valid. Must write 01b to [7:6] bits. The minimum value is 2. | 42h |

Clock configuration register (R8_CLK_CFG_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:2] | Reserved | RWA | Reserved. Must write 01b to [7:6] bits. | 100000b |
| 1 | RB_CLK_SEL_PLL | RWA | System clock source selection.<br>1: 480MHz provided by USB PHY;<br>0: External crystal oscillator 30MHz. | 0 |
| 0 | RB_CLK_PLL_SLEEP | RWA | PLL sleep control.<br>1: PLL sleep;<br>0: PLL works normally. | 0 |

Clock auxiliary register (R8_CLK_MOD_AUX)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RWA | Reserved. | 0 |
| 4 | RB_MCO_EN | RWA | MCO pin output enable<br>1: Enabled; 0: Disabled. | 0 |
| [3:2] | RB_MCO_SEL_MSK | RWA | MCO pin output clock selection<br>00: 125MHz clock;<br>01: 25MHz clock;<br>1x: 2.5MHz clock. | 0 |
| 1 | RB_EXT_125M_EN | RWA | External 125MHz clock used for Ethernet enable<br>1: Enabled; 0: Disabled. | 0 |
| 0 | RB_INT_125M_EN | RWA | USB PHY 125MHz clock used Ethernet enable<br>1: Enabled; 0: Disabled. | 0 |

*Note: Only one clock can be enabled for RB_EXT_125M_EN and RB_INT_125M_EN. To switch, the clock should be disabled and then another clock can be enabled.*

Sleep control register 0 (R8_SLP_CLK_OFF0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_SLP_CLK_UART3 | RWA | UART3 clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 6 | RB_SLP_CLK_UART2 | RWA | UART2 clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 5 | RB_SLP_CLK_UART1 | RWA | UART1 clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 4 | RB_SLP_CLK_UART0 | RWA | UART0 clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 3 | RB_SLP_CLK_PWMX | RWA | PWM clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 2 | RB_SLP_CLK_TMR2 | RWA | TMR2 clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 1 | RB_SLP_CLK_TMR1 | RWA | TMR1 clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 0 | RB_SLP_CLK_TMR0 | RWA | TMR0 clock control.<br>1: Disabled; 0: Enabled. | 0 |

Sleep control register 1 (R8_SLP_CLK_OFF1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_SLP_CLK_DVP | RO | DVP controller clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 6 | RB_SLP_CLK_SERD | RWA | SerDes controller clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 5 | RB_SLP_CLK_USBSS | RWA | USB3.0 controller clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 4 | RB_SLP_CLK_USBHS | RWA | USB2.0 controller clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 3 | RB_SLP_CLK_HSPI | RWA | High-speed parallel interface controller clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 2 | RB_SLP_CLK_EMMC | RWA | EMMC clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 1 | RB_SLP_CLK_SPI1 | RWA | SPI1 clock control.<br>1: Disabled; 0: Enabled. | 0 |
| 0 | RB_SLP_CLK_SPI0 | RWA | SPI0 clock control.<br>1: Disabled; 0: Enabled. | 0 |

*Note: CH569 has no DVP function, so the RB_SLP_CLK_DVP control bit is reserved and writing is invalid. CH565 has no HSPI function, so the RB_SLP_CLK_HSPI control bit is reserved and writing is invalid.*

Wake-up control register (R8_SLP_WAKE_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:6] | Reserved | RO | Reserved. | 0 |
| 5 | RB_SLP_ETH_WAKE | RWA | Ethernet wakeup enable control in low power mode.<br>1: Ethernet wakeup enabled;<br>0: Ethernet wakeup disabled. | 0 |
| 4 | RB_SLP_GPIO_WAKE | RWA | GPIO port wakeup enable control in low power mode.<br>1: GPIO wakeup enabled;<br>0: GPIO wakeup disabled. | 0 |
| 3 | RB_SLP_CLK_ECDC | RWA | Encryption module clock disable control.<br>1: Encryption module clock disabled;<br>0: Encryption module clock enabled. | 0 |
| 2 | RB_SLP_CLK_ETH | RWA | Ethernet clock disable control.<br>1: Ethernet clock disabled;<br>1: Ethernet clock enabled. | 0 |
| 1 | RB_SLP_USBSS_WAKE | RWA | USB3.0 wakeup enable control in low power mode.<br>1: USB3.0 wakeup enabled;<br>0: USB3.0 wakeup disabled. | 0 |
| 0 | RB_SLP_USBHS_WAKE | RWA | USB2.0 wakeup enable control in low power mode.<br>1: USB2.0 wakeup enabled;<br>0: USB2.0 wakeup disabled. | 0 |

Low power control register (R8_SLP_POWER_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:1] | Reserved | R0 | Reserved. | 0 |
| 0 | RB_SLP_USBHS_PWRDN | RWA | USBHS module power control.<br>1: Power down;<br>0: Power on normally. | 0 |

SerDes PHY analog parameter configuration register 1 (R16_SERD_ANA_CFG1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:10] | Reserved | R0 | Reserved. | 0 |
| 9 | RB_SERD_DN_TST | RWA | GXM pin test enable of SerDes PHY.<br>1: Internal test mode. GXM outputs internal voltage value and is no longer used as a serial port for data reception/transmission;<br>0: SerDes working mode. GXM serves as a serial port for data reception/transmission. | 0 |
| 8 | RB_SERD_30M_SEL | RWA | Reference clock source selection of SerDes PHY.<br>1: USB PHY 480MHz divided by 16 selected as the reference clock;<br>0: 30MHz crystal oscillator selected as the reference clock. | 0 |

| | | | | |
|---|---|---|---|---|
| [7:0] | RB_SERD_PLL_CFG | RWA | SerDes PHY built-in PLL configuration bits | 5Ah |

SerDes PHY analog parameter configuration register 2 (R32_SERD_ANA_CFG2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:25] | Reserved | R0 | Reserved. | 0 |
| [24:0] | RB_SERD_TRX_CFG | RWA | Receive/transmit parameter configuration of SerDes PHY.<br>The specific configuration value needs to be adjusted according to the transfer speed. Please refer to the configuration provided by SerDes software routines. | 423015h |

# Chapter 4 Programmable Fast Interrupt Controller (PFIC)

The system has a built-in Programmable Fast Interrupt Controller (PFIC), which supports up to 255 interrupt vectors. The current system manages 21 peripheral interrupt channels and 5 core interrupt channels, and other interrupt sources are reserved.

## 4.1 Main features

- 21+3 individually maskable interrupts. Each interrupt request has an independent trigger bit and mask bit and status bit
- A non-maskable interrupt (NMI)
- Special fast interrupt entry and exit mechanism, hardware automatic stack and resume, without instruction overhead
- Special fast interrupt response mechanism, 4 programmable direct interrupt vector addresses

## 4.2 System timer

The core provides a 64-bit downcounter (SysTick) that supports HCLK or HCLK/8 as the time base. It has a higher priority and can be used as a time reference after calibration.

## 4.3 Interrupt and exception vector

Table 4-1 Vector table

| No. | Priority | Type | Name | Description | Entry address |
|---|---|---|---|---|---|
| 0 | | - | - | | 0x00000000 |
| 1 | -3 | Fixed | Reset | Reset | 0x00000004 |
| 2 | -2 | Fixed | NMI | Non-maskable interrupt | 0x00000008 |
| 3 | -1 | Fixed | EXC | Exception interrupt | 0x0000000C |
| 4-11 | - | - | - | Reserved | |
| 12 | 0 | Programmable | SysTick | System timer interrupt | 0x00000030 |
| 13 | - | - | - | Reserved | |
| 14 | 1 | Programmable | SWI | Software interrupt | 0x00000038 |
| 15 | - | - | - | Reserved | |
| 16 | 2 | Programmable | WDOG | Watchdog timeout reset interrupt | 0x00000040 |
| 17 | 3 | Programmable | TMR0 | Timer0 interrupt | 0x00000044 |
| 18 | 4 | Programmable | GPIO | GPIO port interrupt | 0x00000048 |
| 19 | 5 | Programmable | SPI0 | SPI0 interrupt | 0x0000004C |
| 20 | 6 | Programmable | USBSS | USB3.0 interrupt | 0x00000050 |
| 21 | 7 | Programmable | LINK | USB3.0 link layer interrupt | 0x00000054 |
| 22 | 8 | Programmable | TMR1 | Timer1 interrupt | 0x00000058 |
| 23 | 9 | Programmable | TMR2 | Timer2 interrupt | 0x0000005C |
| 24 | 10 | Programmable | UART0 | UART0 interrupt | 0x00000060 |
| 25 | 11 | Programmable | USBHS | USB2.0 interrupt | 0x00000064 |
| 26 | 12 | Programmable | EMMC | EMMC interrupt | 0x00000068 |
| 27 | 13 | Programmable | DVP | DVP interrupt | 0x0000006C |

| | | | | | |
|---|---|---|---|---|---|
| 28 | 14 | Programmable | HSPI | HSPI interrupt | 0x00000070 |
| 29 | 15 | Programmable | SPI1 | SPI1 interrupt | 0x00000074 |
| 30 | 16 | Programmable | UART1 | UART1 interrupt | 0x00000078 |
| 31 | 17 | Programmable | UART2 | UART2 interrupt | 0x0000007C |
| 32 | 18 | Programmable | UART3 | UART3 interrupt | 0x00000080 |
| 33 | 19 | Programmable | SerDes | SerDes controller interrupt | 0x00000084 |
| 34 | 20 | Programmable | ETH | Ethernet interrupt | 0x00000088 |
| 35 | 21 | Programmable | PMT | Ethernet power management interrupt | 0x0000008C |
| 36 | 22 | Programmable | ECDC | Encryption module interrupt | 0x00000090 |

## 4.4 Register description

### 4.4.1 PFIC register description

PFIC register physical base address: 0xE000E000

Table 4-2 PFIC registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R32_PFIC_ISR1 | 0x00 | PFIC interrupt enable status register 1 | 0x00000000 |
| R32_PFIC_ISR2 | 0x04 | PFIC interrupt enable status register 2 | 0x00000000 |
| R32_PFIC_IPR1 | 0x20 | PFIC interrupt suspend status register 1 | 0x00000000 |
| R32_PFIC_IPR2 | 0x24 | PFIC interrupt suspend status register 2 | 0x00000000 |
| R32_PFIC_ITHRESDR | 0x40 | PFIC interrupt priority threshold configuration register | 0x00000000 |
| R32_PFIC_FIBADDRR | 0x44 | PFIC fast interrupt service base address register | 0x00000000 |
| R32_PFIC_CFGR | 0x48 | PFIC interrupt configuration register | 0x00000000 |
| R32_PFIC_GISR | 0x4C | PFIC interrupt global status register | 0x00000000 |
| R32_PFIC_FIOFADDRR0 | 0x60 | PFIC fast interrupt 0 offset address register | 0x00000000 |
| R32_PFIC_FIOFADDRR1 | 0x64 | PFIC fast interrupt 1 offset address register | 0x00000000 |
| R32_PFIC_FIOFADDRR2 | 0x68 | PFIC fast interrupt 2 offset address register | 0x00000000 |
| R32_PFIC_FIOFADDRR3 | 0x6C | PFIC fast interrupt 3 offset address register | 0x00000000 |
| R32_PFIC_IENR1 | 0x100 | PFIC interrupt enable setting register 1 | 0x00000000 |
| R32_PFIC_IENR2 | 0x104 | PFIC interrupt enable setting register 2 | 0x00000000 |
| R32_PFIC_IRER1 | 0x180 | PFIC interrupt enable reset register 1 | 0x00000000 |
| R32_PFIC_IRER2 | 0x184 | PFIC interrupt enable reset register 2 | 0x00000000 |
| R32_PFIC_IPSR1 | 0x200 | PFIC interrupt suspend setting register 1 | 0x00000000 |
| R32_PFIC_IPSR2 | 0x204 | PFIC interrupt suspend setting register 2 | 0x00000000 |
| R32_PFIC_IPRR1 | 0x280 | PFIC interrupt suspend reset register 1 | 0x00000000 |
| R32_PFIC_IPRR2 | 0x284 | PFIC interrupt suspend reset register 2 | 0x00000000 |
| R32_PFIC_IACTR1 | 0x300 | PFIC interrupt activation status register 1 | 0x00000000 |
| R32_PFIC_IACTR2 | 0x304 | PFIC interrupt activation status register 2 | 0x00000000 |
| R32_PFIC_IPRIORx | 0x400 | PFIC interrupt priority configuration register | 0x00000000 |
| R32_PFIC_SCTLR | 0xD10 | PFIC system control register | 0x00000000 |

PFIC interrupt enable status register 1 (PFIC_ISR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:12] | INTSTA | RO | 31# and below interrupt current enable status.<br>1: Current number interrupt enabled;<br>0: Current number interrupt disabled. | 0 |
| [11:0] | Reserved | RO | Reserved.<br>Reset, NMI and EXC interrupts are ignored, the same below. | 0 |

PFIC interrupt enable status register 2 (PFIC_ISR2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:28] | Reserved | RO | Reserved. | 0 |
| [27:0] | INTENSTA | RO | 32# and above interrupt current enable status.<br>1: Current number interrupt enabled;<br>0: Current number interrupt disabled. | 0 |

PFIC interrupt suspend status register 1 (PFIC_IPR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:12] | PENDSTA | RO | 31# and below interrupt current suspend status.<br>1: Current number interrupt suspended;<br>0: Current number interrupt not suspended; | 0 |
| [11:0] | Reserved | RO | Reserved. | 0 |

PFIC interrupt suspend status register 2 (PFIC_IPR2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:28] | Reserved | RO | Reserved. | 0 |
| [27:0] | PENDSTA | RO | 32# and above interrupt current suspend status.<br>1: Current number interrupt suspended;<br>0: Current number interrupt not suspended; | 0 |

PFIC interrupt priority threshold configuration register (PFIC_ITHRESDR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:8] | Reserved | RO | Reserved. | 0 |
| [7:0] | THRESHOLD | RW | Interrupt priority threshold setting value.<br>If the priority value of the interrupt is lower than the current setting value, interrupt service is not executed when suspended. When this register is 0, the threshold register function is invalid.<br>[7:4]: Priority threshold.<br>[3:0]: Reserved. Fixed to 0. Invalid if writing. | 0 |

PFIC fast interrupt service base address register (PFIC_FIBADDRR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:28] | BASEADDR | RW | The higher 4 bits of target jump address of fast interrupt response.<br>Together with PFIC_FIOFADDRR*, it forms the corresponding-number fast interrupt vector (the 32-bit jump address of the interrupt service program). | 0 |
| [27:0] | Reserved | RO | Reserved. | 0 |

PFIC interrupt configuration register (PFIC_CFGR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:16] | KEYCODE | WO | Correspond to different target control bits, the corresponding safe access identification data needs to be written synchronously for modification. The read data is fixed to 0.<br>KEY1 = 0xFA05;<br>KEY2 = 0xBCAF;<br>KEY3 = 0xBEEF. | 0 |
| [15:8] | Reserved | RO | Reserved. | 0 |
| 7 | SYSRESET | WO | System reset (Write into KEY3 synchronously). Cleared automatically.<br>Valid when writing 1. Invalid when writing 0. | 0 |
| 6 | PFICRESET | WO | PFIC control module reset. Cleared automatically.<br>Valid when writing 1. Invalid when writing 0. | 0 |
| 5 | EXCRESET | WO | Exception interrupt clear suspend (write into KEY2 synchronously).<br>Valid when writing 1. Invalid when writing 0. | 0 |
| 4 | EXCSET | WO | Exception interrupt suspend (write into KEY2 synchronously).<br>Valid when writing 1. Invalid when writing 0. | 0 |
| 3 | NMIRESET | WO | NMI interrupt clear suspend (write into KEY2 synchronously).<br>Valid when writing 1. Invalid when writing 0. | 0 |
| 2 | NMISET | WO | NMI interrupt suspend (write into KEY2 synchronously).<br>Valid when writing 1. Invalid when writing 0. | 0 |
| 1 | NESTCTRL | RW | Nested interrupt enable control:<br>1: Off; 0: On (write into KEY1 synchronously). | 0 |
| 0 | HWSTKCTRL | RW | Hardware stack enable control:<br>1: Off; 0: On (write into KEY1 synchronously). | 0 |

PFIC interrupt global status register (PFIC_GISR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:10] | Reserved | RO | Reserved. | 0 |

| | | | | |
|---|---|---|---|---|
| 9 | GPENDSTA | RO | Whether there is interrupt suspended currently:<br>1: Yes; 0: No. | 0 |
| 8 | GACTSTA | RO | Whether the interrupt is executed currently:<br>1: Yes; 0: No. | 0 |
| [7:0] | NESTSTA | RO | Current interrupt nested status, support 2-level nested currently. The [1:0] bits are valid.<br>3: Level-2 interrupt in process;<br>1: Level-1 interrupt in process;<br>0: No interrupt;<br>Others: Impossible conditions. | 0 |

PFIC fast interrupt 0 offset address register (PFIC_FIOFADDRR0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | IRQID0 | RW | Fast interrupt0 No. | 0 |
| [23:0] | OFFADDR0 | RW | Lower 24-bit address of the fast interrupt 0 service program. Configuration of the lower 20 bits is valid, and [23:20] bits are fixed to 0. | 0 |

PFIC fast interrupt 1 offset address register (PFIC_FIOFADDRR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | IRQID1 | RW | Fast interrupt1 No. | 0 |
| [23:0] | OFFADDR1 | RW | Lower 24-bit address of the fast interrupt 1 service program. Configuration of the lower 20 bits is valid, and [23:20] bits are fixed to 0. | 0 |

PFIC fast interrupt 2 offset address register (PFIC_FIOFADDRR2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | IRQID2 | RW | Fast interrupt2 No. | 0 |
| [23:0] | OFFADDR2 | RW | Lower 24-bit address of the fast interrupt 2 service program. Configuration of the lower 20 bits is valid, and [23:20] bits are fixed to 0. | 0 |

PFIC fast interrupt 3 offset address register (PFIC_FIOFADDRR0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | IRQID3 | RW | Fast interrupt3 No. | 0 |
| [23:0] | OFFADDR3 | RW | Lower 24-bit address of the fast interrupt 3 service program. Configuration of the lower 20 bits is valid, and [23:20] bits are fixed to 0. | 0 |

PFIC interrupt enable setting register 1 (PFIC_IENR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:12] | INTEN | RW1 | 31# and below interrupt enable control.<br>1: Current number interrupt enabled;<br>0: No action. | 0 |
| [11:0] | Reserved | RO | Reserved. | 0 |

PFIC interrupt enable setting register 2 (PFIC_IENR2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:28] | Reserved | RO | Reserved. | 0 |
| [27:0] | INTEN | RW1 | 32# and above interrupt enable control.<br>1: Current number interrupt enabled;<br>0: No action. | 0 |

PFIC interrupt enable reset register 1 (PFIC_IRER1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:12] | INTRESET | RW1 | 31# and below interrupt disable control.<br>1: Current number interrupt disabled;<br>0: No action. | 0 |
| [11:0] | Reserved | RO | Reserved. | 0 |

PFIC interrupt enable reset register 2 (PFIC_IRER2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:28] | Reserved | RO | Reserved. | 0 |
| [27:0] | INTRESET | RW1 | 32# and above interrupt disable control.<br>1: Current number interrupt disabled;<br>0: No action. | 0 |

PFIC interrupt suspend setting register 1 (PFIC_IPSR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:12] | PENDSET | RW1 | 31# and below interrupt suspend setting.<br>1: Current number interrupt suspended;<br>0: No action. | 0 |
| [11:0] | Reserved | RO | Reserved. | 0 |

PFIC interrupt suspend setting register 2 (PFIC_IPSR2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:28] | Reserved | RO | Reserved. | 0 |

| | | | | |
|---|---|---|---|---|
| [27:0] | PENDSET | RW1 | 32# and above interrupt suspend setting.<br>1: Current number interrupt suspended;<br>0: No action. | 0 |

PFIC interrupt suspend reset register 1 (PFIC_IPRR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:12] | PENDRESET | RW1 | 31# and below interrupt suspend reset.<br>1: Current number interrupt reset suspend status;<br>0: No action. | 0 |
| [11:0] | Reserved | RO | Reserved. | 0 |

*Note: The above register is invalid for the interrupts with number of Reset, NMI and EXC.*

PFIC interrupt suspend reset register 2 (PFIC_IPRR2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:28] | Reserved | RO | Reserved. | 0 |
| [27:0] | PENDRESET | RW1 | 32# and above interrupt suspend reset.<br>1: Current number interrupt reset suspend status;<br>0: No action. | 0 |

PFIC interrupt activation status register 1 (PFIC_IACTR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:12] | IACTS | RW1 | 31# and below interrupt activation status.<br>1: Executing the current number interrupt;<br>0: Not executing the current number interrupt. | 0 |
| [11:0] | Reserved | RO | Reserved. | 0 |

PFIC interrupt activation status register 2 (PFIC_IACTR2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:28] | Reserved | RO | Reserved. | 0 |
| [27:0] | IACTS | RW1 | 32# and above interrupt activation status.<br>1: Executing the current number interrupt;<br>0: Not executing the current number interrupt. | 0 |

PFIC interrupt priority configuration register (PFIC_IPRIORx) (x=0-63)

The controller supports 256 interrupts (0-255), and each interrupt uses 8 bits to set the control priority.

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| IPRIOR63 | PRIO_255 | PRIO_254 | PRIO_253 | PRIO_252 |
| … | … | … | … | … |
| IPRIORx | PRIO_(4x+3) | PRIO_(4x+2) | PRIO_(4x+1) | PRIO_(4x) |
| … | … | … | … | … |
| IPRIOR0 | PRIO_3 | PRIO_2 | PRIO_1 | PRIO_0 |

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [2047:2040] | IP_255 | RW | See IP_0 description. | 0 |
| … | … | … | … | … |
| [31:24] | IP_3 | RW | See IP_0 description. | 0 |
| [23:16] | IP_2 | RW | See IP_0 description. | 0 |
| [15:8] | IP_1 | RW | See IP_0 description. | 0 |
| [7:0] | IP_0 | RW | Number 0 interrupt priority configuration:<br>[7:4]: Priority control bit.<br>[3:0]: Reserved. Fixed at 0. Invalid if writing.<br>The smaller priority value means higher priority. There is 2-level interrupt nested, i.e., it can be only seized once. | 0 |

PFIC system control register (PFIC_SCTLR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:6] | Reserved | RO | Reserved. | 0 |
| 5 | SETEVENT | WO | Set event, to wake up the WFE. | 0 |
| 4 | SEVONPEND | RW | When an event/interrupt pending status occurs, the system can be awaken by the WFE command. If the WFE command is not executed, the system is awaken immediately after the next execution of the command.<br>1: Enabled events and all interrupts (including disabled interrupts) can wake up the system;<br>0: Only enabled events and enabled interrupts can wake up the system. | 0 |
| 3 | WFITOWFE | RW | WFI command executed as WFE.<br>1: The subsequent WFI command is executed as WFE command;<br>0: No action. | 0 |
| 2 | SLEEPDEEP | RW | Low power mode of control system:<br>1: deep sleep 0: sleep | 0 |

| | | | | |
|---|---|---|---|---|
| 1 | SLEEPONEXIT | RW | System status after controlled to terminate the interrupt service program:<br>1: System enters the low power mode;<br>0: System enters the main program. | 0 |
| 0 | Reserved | RO | Reserved. | 0 |

### 4.4.2 SysTick register description

STK register physical base address: 0xE000E000

Table 4-3 STK registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R32_STK_CTLR | 0x00 | System count control register | 0x00000000 |
| R32_STK_CNTL | 0x04 | System counter low bit register | 0x00000000 |
| R32_STK_CNTH | 0x08 | System counter high bit register | 0x00000000 |
| R32_STK_CMPLR | 0x0C | Count reload low bit register | 0x00000000 |
| R32_STK_CMPHR | 0x10 | Count reload high bit register | 0x00000000 |
| R32_STK_CNTFG | 0x14 | Counter count flag register | 0x00000000 |

System count control register (STK_CTLR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:9] | Reserved | RO | Reserved. | 0 |
| 8 | STRELOAD | W1 | Reload control, write 1 to update the value of count reload register (64-bit) into the current counter register. | 0 |
| [7:3] | Reserved | RO | Reserved. | 0 |
| 2 | STCLK | RW | Counter clock source selection:<br>1: HCLK selected as the time base;<br>0: HCLK/8 selected as the count time base. | 0 |
| 1 | STIE | RW | Counter interrupt enable control bit:<br>1: Counter interrupt enabled;<br>1: No counter interrupt; | 0 |
| 0 | STE | RW | System counter enable control bit:<br>1: System counter (STK) enabled;<br>0: System counter (STK) disabled. The counter stops counting. | 0 |

System counter low-bit register (STK_CNTL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | CNTL | RW | Lower 32 bits of count value of current counter. Downcount.<br>When CNT[63:0] value reaches 0, the STK interrupt flag is set. | 0 |

System counter high-bit register (STK_CNTH)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | CNTH | RW | Higher 32 bits of count value of current counter. Downcount.<br>When CNT[63:0] value reaches 0, the STK interrupt flag is set. | 0 |

*Note: The STK_CNTL register and the STK_CNTH register consists a 64-bit upcounter.*

Count reload low bit register (STK_CMPLR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | CMPL | RW | Set the reload counter value low 32 bits. | 0 |

Count reload high bit register (STK_CMPHR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | CMPH | RW | Set the reload counter value high 32 bits. | 0 |

*Note: The STK_CMPLR register and the STK_CMPHR register consists the 64-bit counter comparison value.*

Counter count flag register (STK_CNTFG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:2] | Reserved | RO | Reserved. | 0 |
| 1 | CNTIF | RW0 | Flag that counter is decremented to 0. Cleared by writing 0. Invalid when writing 1. | 0 |
| 0 | SWIE | RW | System software interrupt enable bit.<br>1: Software interrupt enabled, at the same time, software interrupt service is triggered;<br>0: Software interrupt disabled. | 0 |

*Note: When SWIE=1, software interrupt service is executed. If the SWIE function is not disabled in the interrupt service, the software interrupt service will be triggered again after leaving the interrupt service.*

# Chapter 5 General Purpose I/O and alternate functions

## 5.1 Introduction to GPIO

The system is equipped with 2 sets of GPIO ports (PA and PB), with a total of 49 general-purpose input/output pins. And some pins have interrupt, alternate and mapping functions.

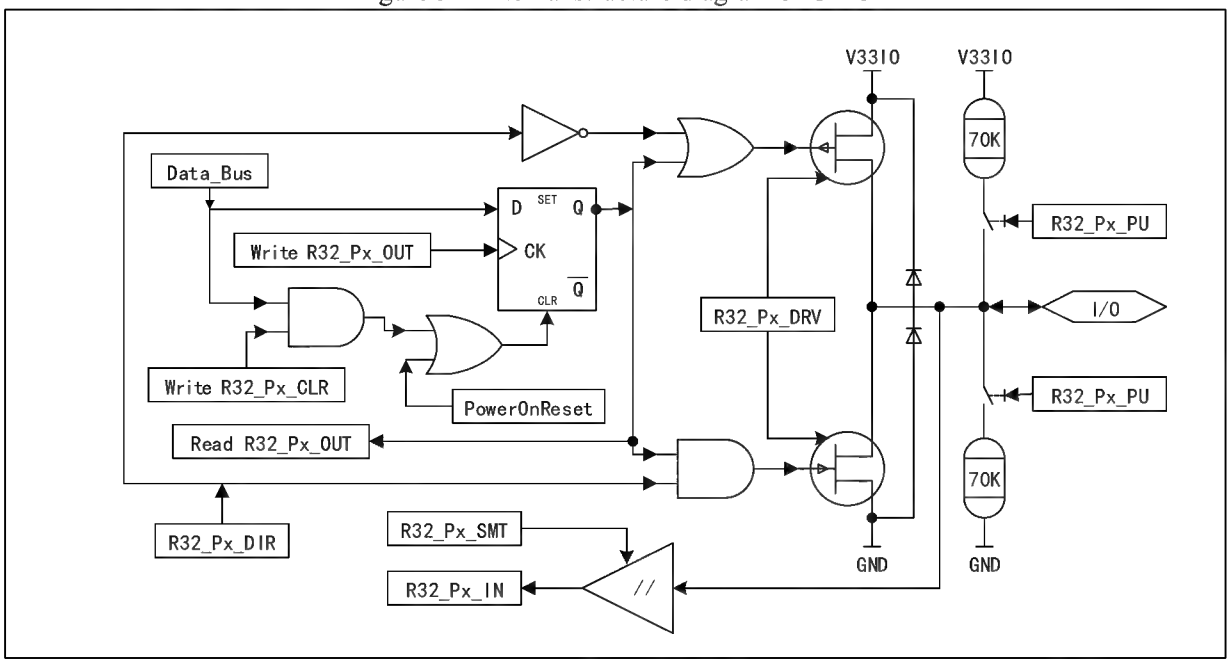
Each GPIO port has the following registers
A 32-bit direction configuration register (R32_Px_DIR)
A 32-bit input data register (R32_Px_PIN)
A 32-bit data output register (R32_Px_OUT)
A32-bit data reset register (R32_Px_CLR)
A 32-bit pull-up configuration register (R32_Px_PU)
A 32-bit open drain output and pull-down enable register (R32_Px_PD)
A 32-bit drive capability configuration register (R32_Px_DRV)
A 32-bit low slope output and Schmitt input register (R32_Px_SMT).

In the PA port, the PA[0]-PA[23] bits are valid, which correspond to the 24 GPIO pins on the chip. In the PB port, the PB[0]-PB[24] bits are valid, which correspond to the 25 GPIO pins on the chip. Among these pins, 8 I/O pins have interrupt function and can implement sleep wakeup function. They are: PA2/PA3/PA4 and PB3/PB4/PB11/PB12/PB15.

Each I/O port bit can be freely programmed, but the I/O port register must be accessed by 8-bit, 16-bit or 32-bit words. If the alternate function of pin is not enabled, it is used as a general-purpose I/O port.

The following figure is a block diagram of the internal structure of GPIO:

Figure 5-1 Internal structure diagram of GPIO




Each GPIO port can be configured into 5 modes, as shown in the table below:

Table 5-1 GPIO pin mode configuration

| PA_SMT | Mode | | Px_DIR | Px_PU | PA_PD | Px_DRV |
|---|---|---|---|---|---|---|
| X | Floating input/high impedance input | | 0 | 0 | 0 | X |
| 1 | No Schmitt trigger | Input with pull-up resistor | 0 | 1 | 0 | X |
| | | Input with pull-down resistor | 0 | 0 | 1 | X |
| 0 | Support Schmitt trigger | Schmitt input with pull-up resistor | 0 | 1 | 0 | X |
| | | Schmitt input with pull-down resistor | 0 | 0 | 1 | X |
| 1 | Low slope | Push-pull output, 8mA drive capability | 1 | X | 0 | 0 |
| | | Push-pull output, 16mA drive capability | 1 | X | 0 | 1 |
| 0 | Quick | Push-pull output, 8mA drive capability | 1 | X | 0 | 0 |
| | | Push-pull output, 16mA drive capability | 1 | X | 0 | 1 |
| X | Open-drain output, 8mA drive capability | | 1 | X | 1 | 0 |
| | Open-drain output, 16mA drive capability | | 1 | X | 1 | 1 |

## 5.2 External interrupt/wakeup

The 8 I/O pins (PA2/PA3/PA4 and PB3/PB4/PB11/PB12/PB15) of the chip have interrupt function and can implement sleep wakeup. In order to use external interrupts, the port bits must be configured in input mode. And 4 kinds of trigger modes are provided: high level, low level, rising edge, falling edge.

The wake-up function needs to turn on the interrupt enable (R8_GPIO_INT_ENABLE) of the port bit, and turn on the GPIO wake-up control bit (RB_SLP_GPIO_WAKE) in the R8_SLP_WAKE_CTRL register.

## 5.3 GPIO alternate functions and remapping

### 5.3.1 Alternate functions

Some I/O pins have alternate functions. After power on, all I/O pins act as general purpose I/O port by default. After different function modules are enabled, the corresponding pins are configured as corresponding function pins of each function module.

If a pin has multiple alternate functions, and multiple functions are enabled, please refer to the function order in the "Alternate and mapping" list in Section 1.3 for the priority order of alternate functions.

For example: PA0 pin is multiplexed as MD5/HD6/BD0, the data line function of EMMC has priority, and the active parallel interface data line has the lowest priority. In this way, the alternate functions with the relatively higher priority of the pin whose functions with the lowest priority need not to be used can be enabled among multiple alternate functions.

The following tables list the I/O pin configuration used by each function module.

Table 5-2 Timer x

| TMR0/1/2/3 pinout | Function configuration | GPIO configuration |
|---|---|---|
| TMRx | Input capture channel x | Input (floating/pull-up/pull-down) |
| | Output PWM channel x | Push-pull output |

Table 5-3 UARTx interface

| UART0/1/2/3 pinout | Function configuration | GPIO configuration |
|---|---|---|
| TXDx | UART transmit x | Push-pull output |
| RXDx | UART receive x | Pull-up input (recommended) or floating input |
| RTS,DTR | MODEM signal input or RS485 control | Push-pull output |
| CTS,DSR,RI,DCD | MODEM signal input | Pull-up input (recommended) or floating input |

Table 5-3 SPIx

| SPI0/1 pinout | Function configuration | GPIO configuration |
|---|---|---|
| SCKx | Master mode clock output | Push-pull output |
| | Slave mode clock input | Input (floating/pull-up/pull-down) |
| MOSIx | Full duplex mode-master mode | Push-pull output |
| | Full duplex mode-slave mode | Input (floating/pull-up/pull-down) |
| | Half duplex mode-master mode | Not used. Can be used as a general I/O |
| | Half duplex mode-slave mode | Not used. Can be used as a general I/O |
| MISOx | Full duplex mode-master mode | Input (floating/pull-up/pull-down) |
| | Full duplex mode-slave mode | Input (pull-up recommended, the hardware automatically switches to push-pull output after chip select) or push-pull output (cannot be used for bus connection) |
| | Half duplex mode-master mode | Input or push-pull output, software switch |
| | Half duplex mode-slave mode | Input (pull-up recommended, hardware automatically switches to push-pull output after chip select) |
| SCS | Master mode chip select output | Push-pull output (can be replaced with other pins) |
| | Slave mode chip select output | Pull-up input (recommended) or floating input |

Table 5-4 High-speed parallel interface

| HSPI pinout | Function configuration | GPIO configuration |
|---|---|---|
| HTCLK,HTREQ,HTVLD, HTACK | Control output signal | Push-pull output |
| HTRDY, HRCLK,HRACT,HRVLD | Control input signal | Pull-down input (recommended) or floating input |
| HD0～HD31 | Data signal | Floating input |

Table 5-5 DVP

| DVP pinout | Function configuration | GPIO configuration |
|---|---|---|
| DVSYNC,DHSYNC | Sync signal input | Pull-up input (recommended) or floating input |
| DPCLK | Clock signal input | Pull-up input (recommended) or floating input |
| DD0～DD31 | Data signal | Pull-up input (recommended) or floating input |

Table 5-6 EMMC interface

| EMMC pinout | Function configuration | GPIO configuration |
|---|---|---|
| MSDCK | Clock signal output | Push-pull output |
| MCMD | Command signal output | Push-pull output |
| MD0～MD7 | Data signal | Pull-up input (recommended) or floating input |

Table 5-7 Ethernet interface

| MII pinout | Function configuration | GPIO configuration |
|---|---|---|
| ETXD0～ETXD3 | Ethernet transmit data signal | Push-pull output |
| ERXD0～ERXD3 | Ethernet receive data signal | Pull-up input (recommended) or floating input |
| ERXDV,ERXC | Control and clock signal input | Pull-up input (recommended) or floating input |
| ETXC,ETXEN | Control and clock signal output | Push-pull output |
| EMDIO | SMI management data | Pull-up input (recommended) or floating input |
| EMDCK | SMI management clock | Push-pull output |
| ETCKI | Ethernet clock input | Pull-up input (recommended) or floating input |
| EMCO | Ethernet clock output | Push-pull output |

Table 5-8 BUS8 interface

| BUS8 pinout | Function configuration | GPIO configuration |
|---|---|---|
| BRD#,BWR# | Bus control | Push-pull output |
| BD0～BD7,BA0～BA14 | Bus data, address | Pull-up input (recommended) or floating input |

Table 5-9 High-speed analog signal interface

| High-speed analog pinout | Function configuration | GPIO configuration |
|---|---|---|
| UD+,UD- | Connected to internal USB PHY | Floating input |
| SSTXA,SSTXB | | Floating input |
| SSRXA,SSRXB | | Floating input |
| GXM,GXP | Connected to internal SerDes PHY | Floating input |

### 5.3.2 Function pin remapping

In order to optimize the simultaneous utilization of peripheral functions, some function's hardware can be remapped to other pins by setting the R8_PIN_ALTERNATE register. The system supports the remapping of UART0, TMR1 and TMR2 peripheral pins, please refer to the following table for details:

Table 5-10 Remapping pins

| **Peripheral function** | **Default pin** | **Remapping pin** |
|---|---|---|
| UART0 | PB5/PB6 | PA5/PA6 |
| TMR1/PWM5 | PB15 | PB0 |
| TMR2/PWM6 | PA4 | PB3 |

## 5.4 Register description

GPIO register physical base address: 0x40001000

Table 5-11 GPIO registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_GPIO_INT_FLAG | 0x1C | GPIO interrupt flag register | 0x00 |
| R8_GPIO_INT_ENABLE | 0x1D | GPIO interrupt enable register | 0x00 |
| R8_GPIO_INT_MODE | 0x1E | GPIO interrupt mode register | 0x00 |
| R8_GPIO_INT_POLAR | 0x1F | GPIO interrupt polarity register | 0x00 |
| R32_PA_DIR | 0x40 | PA port direction setting register | 0x00000000 |
| R32_PA_PIN | 0x44 | PA port data input register | 0x00XXXXXX |
| R32_PA_OUT | 0x48 | PA port data output register | 0x00000000 |
| R32_PA_CLR | 0x4C | PA port data reset register | 0x00000000 |
| R32_PA_PU | 0x50 | PA port pull-up enable register | 0x00000000 |
| R32_PA_PD | 0x54 | PA port open-drain output and input pull-down enable register | 0x00000000 |
| R32_PA_DRV | 0x58 | PA port drive capability configuration register | 0x00000000 |
| R32_PA_SMT | 0x5C | PA port low slope output and Schmitt input register | 0x00000000 |
| R32_PB_DIR | 0x60 | PB port direction setting register | 0x00000000 |
| R32_PB_PIN | 0x64 | PB port data input register | 0x0XXXXXXX |
| R32_PB_OUT | 0x68 | PB port data output register | 0x00000000 |
| R32_PB_CLR | 0x6C | PB port data reset register | 0x00000000 |
| R32_PB_PU | 0x70 | PB port pull-up configuration register | 0x00000000 |
| R32_PB_PD | 0x74 | PB port open-drain output and input pull-down configuration register | 0x00000000 |
| R32_PB_DRV | 0x78 | PB port drive capability configuration register | 0x00000000 |
| R32_PB_SMT | 0x7C | PB port low slope output and Schmitt input register | 0x00000000 |
| R8_PIN_ALTERNATE | 0x12 | Alternate and remapping configuration register | 0x00 |

GPIO interrupt flag register (R8_GPIO_INT_FLAG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_GPIO_PB15_IF | RW1 | PB15 pin interrupt flag bit. Cleared by writing 1.<br>1: Interrupt; 0: No interrupt. | 0 |
| 6 | RB_GPIO_PB12_IF | RW1 | PB12 pin interrupt flag bit. Cleared by writing 1.<br>1: Interrupt; 0: No interrupt. | 0 |
| 5 | RB_GPIO_PB11_IF | RW1 | PB11 pin interrupt flag bit. Cleared by writing 1.<br>1: Interrupt; 0: No interrupt. | 0 |
| 4 | RB_GPIO_PB4_IF | RW1 | PB4 pin interrupt flag bit. Cleared by writing 1.<br>1: Interrupt; 0: No interrupt. | 0 |

| | | | | |
|---|---|---|---|---|
| 3 | RB_GPIO_PB3_IF | RW1 | PB3 pin interrupt flag bit. Cleared by writing 1.<br>1: Interrupt; 0: No interrupt. | 0 |
| 2 | RB_GPIO_PA4_IF | RW1 | PA4 pin interrupt flag bit. Cleared by writing 1.<br>1: Interrupt; 0: No interrupt. | 0 |
| 1 | RB_GPIO_PA3_IF | RW1 | PA3 pin interrupt flag bit. Cleared by writing 1.<br>1: Interrupt; 0: No interrupt. | 0 |
| 0 | RB_GPIO_PA2_IF | RW1 | PA2 pin interrupt flag bit. Cleared by writing 1.<br>1: Interrupt; 0: No interrupt. | 0 |

GPIO interrupt enable register (R8_GPIO_INT_ENABLE)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_GPIO_PB15_IE | RW | PB15 pin interrupt enable.<br>1: Enabled; 0: Disabled. | 0 |
| 6 | RB_GPIO_PB12_IE | RW | PB12 pin interrupt enable.<br>1: Enabled; 0: Disabled. | 0 |
| 5 | RB_GPIO_PB11_IE | RW | PB11 pin interrupt enable.<br>1: Enabled; 0: Disabled. | 0 |
| 4 | RB_GPIO_PB4_IE | RW | PB4 pin interrupt enable.<br>1: Enabled; 0: Disabled. | 0 |
| 3 | RB_GPIO_PB3_IE | RW | PB3 pin interrupt enable.<br>1: Enabled; 0: Disabled. | 0 |
| 2 | RB_GPIO_PA4_IE | RW | PA4 pin interrupt enable.<br>1: Enabled; 0: Disabled. | 0 |
| 1 | RB_GPIO_PA3_IE | RW | PA3 pin interrupt enable.<br>1: Enabled; 0: Disabled. | 0 |
| 0 | RB_GPIO_PA2_IE | RW | PA2 pin interrupt enable.<br>1: Enabled; 0: Disabled. | 0 |

GPIO interrupt mode register (R8_GPIO_INT_MODE)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_GPIO_PB15_IM | RW | PB15 pin interrupt mode selection.<br>1: Edge trigger; 0: Level trigger. | 0 |
| 6 | RB_GPIO_PB12_IM | RW | PB12 pin interrupt mode selection.<br>1: Edge trigger; 0: Level trigger. | 0 |
| 5 | RB_GPIO_PB11_IM | RW | PB11 pin interrupt mode selection.<br>1: Edge trigger; 0: Level trigger. | 0 |
| 4 | RB_GPIO_PB4_IM | RW | PB4 pin interrupt mode selection.<br>1: Edge trigger; 0: Level trigger. | 0 |
| 3 | RB_GPIO_PB3_IM | RW | PB3 pin interrupt mode selection.<br>1: Edge trigger; 0: Level trigger. | 0 |
| 2 | RB_GPIO_PA4_IM | RW | PA4 pin interrupt mode selection.<br>1: Edge trigger; 0: Level trigger. | 0 |
| 1 | RB_GPIO_PA3_IM | RW | PA3 pin interrupt mode selection.<br>1: Edge trigger; 0: Level trigger. | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | RB_GPIO_PA2_IM | RW | PA2 pin interrupt mode selection.<br>1: Edge trigger; 0: Level trigger. | 0 |

GPIO interrupt polarity register (R8_GPIO_INT_POLAR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_GPIO_PB15_IP | RW | PB15 pin interrupt polarity selection.<br>1: High level/rising edge;<br>0: Low level/falling edge. | 0 |
| 6 | RB_GPIO_PB12_IP | RW | PB12 pin interrupt polarity selection.<br>1: High level/rising edge;<br>0: Low level/falling edge. | 0 |
| 5 | RB_GPIO_PB11_IP | RW | PB11 pin interrupt polarity selection.<br>1: High level/rising edge;<br>0: Low level/falling edge. | 0 |
| 4 | RB_GPIO_PB4_IP | RW | PB4 pin interrupt polarity selection.<br>1: High level/rising edge;<br>0: Low level/falling edge. | 0 |
| 3 | RB_GPIO_PB3_IP | RW | PB3 pin interrupt polarity selection.<br>1: High level/rising edge;<br>0: Low level/falling edge. | 0 |
| 2 | RB_GPIO_PA4_IP | RW | PA4 pin interrupt polarity selection.<br>1: High level/rising edge;<br>0: Low level/falling edge. | 0 |
| 1 | RB_GPIO_PA3_IP | RW | PA3 pin interrupt polarity selection.<br>1: High level/rising edge;<br>0: Low level/falling edge. | 0 |
| 0 | RB_GPIO_PA2_IP | RW | PA2 pin interrupt polarity selection.<br>1: High level/rising edge;<br>0: Low level/falling edge. | 0 |

PA port direction setting register (R32_PA_DIR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | Reserved | R0 | Reserved. | 0 |
| [23:0] | R32_PA_DIR | RW | Current input/output direction of PA pin.<br>1: Output; 0: Input. | 0 |

PA port input data register (R32_PA_PIN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | Reserved | R0 | Reserved. | 0 |
| [23:0] | R32_PA_PIN | RO | Current level status of PA pin:<br>1: High level; 0: Low level.<br>*Note: The value of these bits is valid only when the corresponding bit in the direction register (R32_PA_DIR) is 0.* | 0 |

PA port output data register (R32_PA_OUT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | Reserved | R0 | Reserved. | 0 |
| [23:0] | R32_PA_OUT | RW | PA pin output level status.<br>1: Output high level; 0: Output low level.<br>*Note: The value of these bits is valid only when the corresponding bit in the direction register (R32_PA_DIR) is 1.* | 0 |

PA port data reset register (R32_PA_CLR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | Reserved | R0 | Reserved. | 0 |
| [23:0] | R32_PA_CLR | WZ | PA pin output low level.<br>1: Output low level; 0: No action.<br>*Note: When a bit is set to 1, the corresponding bit in the R32_PA_OUT register is cleared.* | 0 |

PA port pull-up configuration register (R32_PA_PU)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | Reserved | R0 | Reserved. | 0 |
| [23:0] | R32_PA_PU | RW | PA pin pull-up function enable.<br>1: Pull-up enabled;<br>0: Pull-up disabled. | 0 |

PA port open-drain output and input pull-down enable register (R32_PA_PD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | Reserved | R0 | Reserved. | 0 |
| [23:0] | R32_PA_PD | RW | When the pin direction is output:<br>1: Open-drain output enabled;<br>0: Open-drain output disabled.<br>When the pin direction is input:<br>1: Pull-down enabled;<br>0: Pull-down disabled. | 0 |

PA port drive capability configuration register (R32_PA_DRV)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | Reserved | R0 | Reserved. | 0 |
| [23:0] | R32_PA_DRV | RW | PA pin output drive capability.<br>1: The maximum driving current is 16mA;<br>0: The maximum drive current is 8mA. | 0 |

PA port low slope output and Schmitt input register (R32_PA_SMT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | Reserved | R0 | Reserved. | 0 |
| [23:0] | R32_PA_SMT | RW | When the pin direction is output:<br>1: Low slope output enabled;<br>0: Low slope output disabled.<br>When the pin direction is input:<br>1: Schmitt trigger input enabled;<br>0: Schmitt trigger input disabled. | 1 |

PB port direction setting register (R32_PB_DIR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:25] | Reserved | R0 | Reserved. | 0 |
| [24:0] | R32_PB_DIR | RW | Current input/output direction of PB pin.<br>1: Output; 0: Input. | 0 |

PB port input data register (R32_PB_PIN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:25] | Reserved | R0 | Reserved. | 0 |
| [24:0] | R32_PB_PIN | RO | Current level status of PB pin.<br>1: High level; 0: Low level.<br>*Note: The value of these bits is valid only when the corresponding bit in the direction register (R32_PB_DIR) is 0.* | 0 |

PB port output data register (R32_PB_OUT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:25] | Reserved | R0 | Reserved. | 0 |
| [24:0] | R32_PB_OUT | RW | PB pin output level status:<br>1: Output high level; 0: Output low level.<br>*Note: The value of these bits is valid only when the corresponding bit in the direction register (R32_PB_DIR) is 1.* | 0 |

PB port data reset register (R32_PB_CLR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:25] | Reserved | R0 | Reserved. | 0 |
| [24:0] | R32_PB_CLR | WZ | PB pin output low level.<br>1: Output low level; 0: No action.<br>*Note: When a bit is set to 1, the corresponding bit in the R32_PB_OUT register is cleared.* | 0 |

PB port pull-up configuration register (R32_PB_PU)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:25] | Reserved | R0 | Reserved. | 0 |
| [24:0] | R32_PB_PU | RW | PB pin pull-up function enable.<br>1: Pull-up enabled;<br>0: Pull-up disabled. | 0 |

PB port open-drain output and input pull-down enable register (R32_PB_PD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:25] | Reserved | R0 | Reserved. | 0 |
| [24:0] | R32_PB_PD | RW | When the pin direction is output:<br>1: Open-drain output enabled;<br>0: Open-drain output disabled.<br>When the pin direction is input:<br>1: Pull-down enabled;<br>0: Pull-down disabled. | 0 |

PB port drive capability configuration register (R32_PB_DRV)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:25] | Reserved | R0 | Reserved. | 0 |
| [24:0] | R32_PB_DRV | RW | PB pin output drive capability.<br>1: The maximum driving current is 16mA;<br>0: The maximum drive current is 8mA. | 0 |

PB port low slope output and Schmitt input register (R32_PB_SMT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:25] | Reserved | R0 | Reserved. | 0 |
| [24:0] | R32_PB_SMT | RW | When the pin direction is output:<br>1: Low slope output enabled;<br>0: Low slope output disabled.<br>When the pin direction is input:<br>1: Schmitt trigger input enabled;<br>0: Schmitt trigger input disabled. | 1 |

Alternate remapping configuration register (R8_PIN_ALTERNATE)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RO | Reserved. | 0 |
| 4 | RB_PIN_UART0 | RW | UART0 remapping configuration.<br>1: RXD0/TXD0 to PA5/PA6 pin;<br>0: RXD0/TXD0 to PB5/PB6 pin. | 0 |

| | | | | |
|---|---|---|---|---|
| 3 | Reserved | RO | Reserved. | 0 |
| 2 | RB_PIN_TMR2 | RW | TIMER2 remapping configuration.<br>1: TMR2/PWM6/CAP2 to PB3 pin;<br>0: TMR2/PWM6/CAP2 to PA4 pin. | 0 |
| 1 | RB_PIN_TMR1 | RW | TIMER1 remapping configuration.<br>1: TMR1/PWM5/CAP1 to PB0 pin;<br>0: TMR1/PWM5/CAP1 to PB15 pin. | 0 |
| 0 | RB_PIN_MII | RW | Ethernet transceiver interface configuration.<br>1: RGMII interface;<br>0: RMII interface. | 1 |

# Chapter 6 Serial peripheral interface (SPI)

SPI is a full-duplex serial interface. A master and several slaves are connected to the bus, and only a pair of master and slave can communicate at the same time. Usually SPI interface consists of 4 pins: SPI Chip Select pin (SCS), SPI Clock pin (SCK), SPI serial data pin MISO (master input/slave output pin) and SPI serial data pin MOSI (master output/slave input pin).

## 6.1 Main features

The system is provided with 2 SPI interfaces, and their functions and operations are the same.

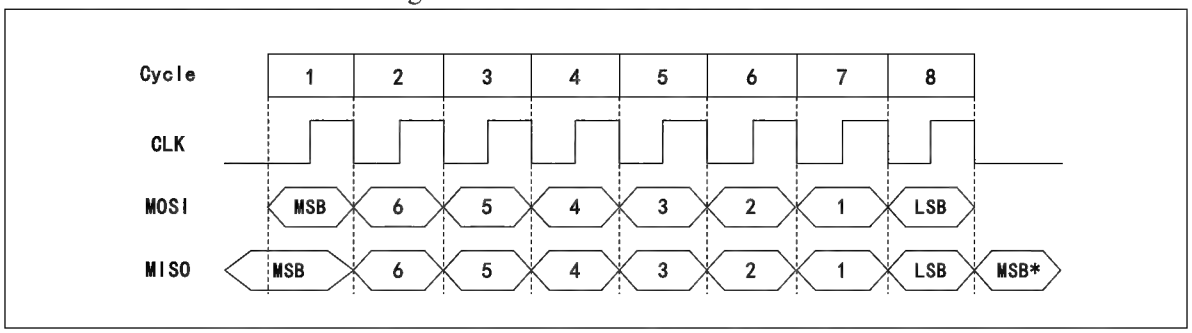
- Support master mode and slave mode
- Compliant with Serial Peripheral Interface (SPI) Specification
- Support 2 transfer frame formats: mode0 and mode3
- 8-bit data transfer mode. Programmable data order with MSB-first or LSB-first shifting
- The maximum clock frequency is close to half of Fsys
- 8-byte FIFO space, multiple transmission flag notification
- The first byte is in command mode or data stream mode in slave mode
- DMA capability for transmission and reception

## 6.2 SPI data frame format

SPI supports 2 transfer frame formats (mode0 and mode3). It can be selected by setting the RB_SPI_MST_SCK_MOD bit in the R8_SPIx_CTRL_MOD register.

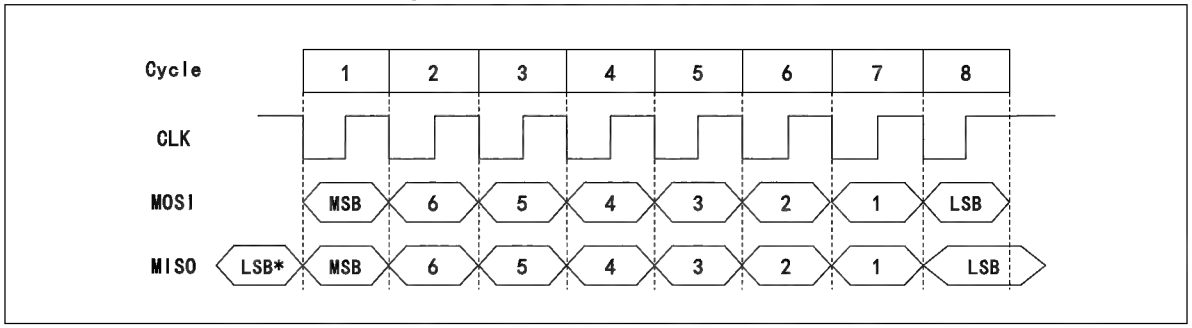
Mode0: The clock idle level is low, and sampling starts on the first edge of the clock (rising edge sampling). RB_SPI_MST_SCK_MOD=0

Figure 6-1 SPI mode0 transfer format




Mode 3: The clock idle level is high, and sampling starts on the second edge of the clock (rising edge sampling). RB_SPI_MST_SCK_MOD=1

Figure 6-2 SPI mode3 transfer format




## 6.3 SPI functional description

### 6.3.1 Master mode

In the master configuration, the serial clock is generated on the SCK pin, and the chip select pin can be specified as any I/O pin.

Procedure:

1. Set the SPI master mode clock divider register (R8_SPIx_CLOCK_DIV), to configure SPI clock speed;
2. Set the RB_SPI_MODE_SLAVE bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD) to 0, to configure SPI in master mode;
3. Set the RB_SPI_MST_SCK_MOD bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD), to select mode0 or mode3 according to the requirements of the connected device;
4. Set the RB_SPI_FIFO_DIR bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD), to configure the FIFO direction. If it is 1, the current FIFO direction is input. If it is 0, the current FIFO direction is output.
5. Set the RB_SPI_MOSI_OE bit and RB_SPI_SCK_OE bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD) to 1, and set the RB_SPI_MISO_OE bit to 0, and set the bits corresponding to the MOSI pin and SCK pin in the PA port direction register (R32_PB_DIR) to 1, set the bit corresponding to MISO pin to 0, to configure the MOSI pin and SCK pin direction as output, and configure the MISO pin direction as input;

Data transmission:

1. Set the RB_SPI_FIFO_DIR bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD) to 0, to configure FIFO direction as output;
2. Write to the R16_SPIx_TOTAL_CNT register, to set the length of the data to be transmitted;
3. Write to the R8_SPIx_FIFO register, to write the data to be transmitted into FIFO. If R8_SPIx_FIFO_COUNT is less than FIFO size, continue to write data to FIFO;
4. After all data are written to FIFO, wait until the R16_SPIx_TOTAL_CNT register becomes 0, indicating that data transmission is completed. If only one byte is sent, you can also wait until R8_SPIx_FIFO_COUNT becomes 0, indicating that there is no data in FIFO and the data transmission is completed.

Data reception:

1. Set the RB_SPI_FIFO_DIR bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD) to 1, to configure FIFO direction as input;
2. Write to the R16_SPIx_TOTAL_CNT register, to set the length of the data to be received;

3. Wait until the R8_SPIx_FIFO_COUNT register is not 0, indicating that the return data is received;
4. The value read in R8_SPIx_FIFO is the received data.

### 6.3.2 Slave mode

In the slave configuration, the serial clock is received on the SCK pin from the connected SPI master device, and the hardware chip select pin needs to be selected.

Procedure:

1. Set the RB_SPI_MODE_SLAVE bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD) to 1, to configure SPI as the slave mode;
2. Set the RB_SPI_SLV_CMD_MOD bit in R8_SPIx_CTRL_MOD as needed, to select the first byte mode or data stream mode of slave;
3. Set the RB_SPI_FIFO_DIR bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD), to configure the FIFO direction. If it is 1, the current FIFO direction is input. If it is 0, the current FIFO direction is output.
4. Set RB_SPI_MOSI_OE and RB_SPI_SCK_OE in R8_SPI0_CTRL_MOD to 0, set RB_SPI_MISO_OE to 1, and set the GPIO direction configuration register (R32_PA/PB_DIR) to configure the MOSI pin, SCK pin and SCS pin as input, and to configure the MISO pin as input (Multiple slaves can be connected on the bus. MISO is automatically switched to output after chip select. Also support one master with one slave) or output (only for connection between one master and one slave). In SPI slave mode, the I/O pin direction of MISO can be configures as output by the GPIO direction configuration register, as well as can be automatically switched to output during valid SPI chip select. While its output data is selected by RB_SPI_MISO_OE. If it is 1, SPI data is output. If it is 0, data in the GPIO data output register is output. It is recommended to set the MISO pin as input so that MISO does not output when chip select is invalid, so that SPI bus can be shared when multiple devices are operated;
5. Optionally, set the SPI0 slave mode preset data register (R8_SPI0_SLAVE_PRE), to be automatically loaded into the buffer for the first time after chip select for external output. After 8 clocks (that is, the exchange of the first data byte between the master and the slave is completed), the controller obtains the first byte of data (command code) sent by the external SPI master, and the external SPI master obtains the preset data (status value) in R8_SPI0_SLAVE_PRE through exchange. The bit7 in R8_SPI0_SLAVE_PRE is automatically loaded into the MISO pin during low SCK after the SPI chip select is valid. For SPI mode0 (CLK defaults to low), if the bit7 in R8_SPI0_SLAVE_PRE is preset, the external SPI master obtains the preset value of bit7 in R8_SPI0_SLAVE_PRE by inquiring the MISO pin when the SPI chip select is valid but there is no data transmission, thereby the value of bit7 in R8_SPI0_SLAVE_PRE can be obtained only by valid SPI chip select (Usually a busy state is provided to the master in order to query quickly);
6. Optional step. If DMA is enabled, write the start address of transceiver buffer to R16_SPI_DMA_BEG and the end address (not included) to R16_SPI_DMA_END. It is recommended to set RB_SPI_DMA_ENABLE after RB_SPI_FIFO_DIR is set.

Data transmission:

1. Set the RB_SPI_FIFO_DIR bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD) to 0, to configure FIFO direction as output;
2. Write the data to be transmitted into FIFO register (R8_SPIx_FIFO), to decide when to take it using the external SPI master. If DMA is enabled, DMA automatically loads FIFO to complete this step;

3. Query R8_SPI0_FIFO_COUNT. If it is not full, continue to write data to be transmitted to FIFO.

Data reception:

1. Set the RB_SPI_FIFO_DIR bit in the SPI mode configuration register (R8_SPIx_CTRL_MOD) to 1, to configure FIFO direction as input;
2. Query R8_SPI0_FIFO_COUNT. If it is not empty, the data has been received, and the data can be taken by reading R8_SPI0_FIFO. If DMA is enabled, DMA automatically reads FIFO to complete this step;
3. For reception of single byte data, FIFO is not necessary, and R8_SPI0_BUFFER can be read directly.

## 6.4 Register description

SPI0 register physical base address: 0x40004000

SPI1 register physical base address: 0x40004400

Table 6-1 SPI0 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_SPI0_CTRL_MOD | 0x00 | SPI0 mode configuration register | 0x02 |
| R8_SPI0_CTRL_CFG | 0x01 | SPI0 configuration register | 0x00 |
| R8_SPI0_INTER_EN | 0x02 | SPI0 interrupt enable register | 0x00 |
| R8_SPI0_CLOCK_DIV<br>R8_SPI0_SLAVE_PRE | 0x03 | SPI0 master mode clock divider register<br>SPI0 slave mode preset data register | 0x10 |
| R8_SPI0_BUFFER | 0x04 | SPI0 data buffer | 0xXX |
| R8_SPI0_RUN_FLAG | 0x05 | SPI0 running flag register | 0x00 |
| R8_SPI0_INT_FLAG | 0x06 | SPI0 interrupt flag register | 0x00 |
| R8_SPI0_FIFO_COUNT | 0x07 | SPI0 transceiver FIFO count register | 0x00 |
| R16_SPI0_TOTAL_CNT | 0x0C | SPI0 transceiver data length register | 0x0000 |
| R8_SPI0_FIFO | 0x10 | SPI0 FIFO register | 0xXX |
| R8_SPI0_FIFO_COUNT1 | 0x13 | SPI0 transceiver FIFO count register | 0x00 |
| R32_SPI0_DMA_NOW | 0x14 | Current address of SPI0 DMA buffer | 0xXXXX |
| R32_SPI0_DMA_BEG | 0x18 | Start address of SPI0 DMA buffer | 0xXXXX |
| R32_SPI0_DMA_END | 0x1C | End address of SPI0 DMA buffer | 0xXXXX |

Table 6-2 SPI1 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_SPI1_CTRL_MOD | 0x00 | SPI1 mode configuration register | 0x02 |
| R8_SPI1_CTRL_CFG | 0x01 | SPI1 configuration register | 0x00 |
| R8_SPI1_INTER_EN | 0x02 | SPI1 interrupt enable register | 0x00 |
| R8_SPI1_CLOCK_DIV<br>R8_SPI1_SLAVE_PRE | 0x03 | SPI1 master mode clock divider register<br>SPI1 slave mode preset data register | 0x10 |
| R8_SPI1_BUFFER | 0x04 | SPI1 data buffer | 0xXX |
| R8_SPI1_RUN_FLAG | 0x05 | SPI1 running flag register | 0x00 |
| R8_SPI1_INT_FLAG | 0x06 | SPI1 interrupt flag register | 0x00 |
| R8_SPI1_FIFO_COUNT | 0x07 | SPI1 transceiver FIFO count register | 0x00 |

| | | | |
|---|---|---|---|
| R16_SPI1_TOTAL_CNT | 0x0C | SPI1 transceiver data length register | 0x0000 |
| R8_SPI1_FIFO | 0x10 | SPI1 FIFO register | 0xXX |
| R8_SPI1_FIFO_COUNT1 | 0x13 | SPI1 transceiver FIFO count register | 0x00 |
| R32_SPI1_DMA_NOW | 0x14 | Current address of SPI1 DMA buffer | 0xXXXX |
| R32_SPI1_DMA_BEG | 0x18 | Start address of SPI1 DMA buffer | 0xXXXX |
| R32_SPI1_DMA_END | 0x1C | End address of SPI1 DMA buffer | 0xXXXX |

SPI mode configuration register (R8_SPIx_CTRL_MOD) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_SPI_MISO_OE | RW | MISO pin output enable (can be used in 2-wire mode for data line direction switch).<br>1: MISO pin output enabled;<br>0: MISO pin output disabled. | 0 |
| 6 | RB_SPI_MOSI_OE | RW | MOSI pin output enable.<br>1: MOSI pin output enabled;<br>0: MOSI pin output disabled. | 0 |
| 5 | RB_SPI_SCK_OE | RW | SCK pin output enable.<br>1: SCK pin output enabled;<br>0: SCK pin output disabled. | 0 |
| 4 | RB_SPI_FIFO_DIR | RW | FIFO direction selection.<br>1: Input (to receive data);<br>0: Output (to transmit data). | 0 |
| 3 | RB_SPI_SLV_CMD_MOD | RW | SPI slave mode first byte mode selection.<br>1: First byte command mode;<br>0: Data flow mode.<br>In the first byte command mode, it is regarded as a command code when the first byte of data is received after the SPI chip select is valid, and the RB_SPI_IF_FST_BYTE bit in the flag register is set to 1.<br>This bit is only valid in slave mode. | 0 |
| | RB_SPI_MST_SCK_MOD | RW | Clock idle mode selection in master mode.<br>1: Mode3 (SCK is at high level when idle);<br>0: Mode0 (SCK is at low level when idle).<br>This bit is only valid in master mode. | 0 |
| 2 | RB_SPI_2WIRE_MOD | RW | 2-wire or 3-wire SPI mode configuration bit in slave mode:<br>1: 2-wire mode/half duplex (SCK, MISO);<br>0: 3-wire mode/full duplex (SCK, MOSI, MISO). | 0 |
| 1 | RB_SPI_ALL_CLEAR | RW | FIFO/counter/interrupt flag register clear.<br>1: Forced to be empty or cleared;<br>0: Not cleared. | 1 |
| 0 | RB_SPI_MODE_SLAVE | RW | SPI master/slave mode selection bit:<br>1: Slave mode; 0: Master mode. | 0 |

SPI configuration register (R8_SPIx_CTRL_CFG) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:6] | Reserved | R0 | Reserved. | 0 |
| 5 | RB_SPI_BIT_ORDER | RW | SPI data bit order selection.<br>1: LSB first; 0: HSB first. | 0 |
| 4 | RB_SPI_AUTO_IF | RW | RB_SPI_IF_BYTE_END flag bit auto clear enable when accessing BUFFER/FIFO.<br>1: Enabled; 0: Disabled. | 0 |
| 3 | Reserved | R0 | Reserved. | 0 |
| 2 | RB_SPI_DMA_LOOP | RW | DMA address loop enable.<br>1: Loop; 0: Once. | 0 |
| 1 | Reserved | R0 | Reserved. | 0 |
| 0 | RB_SPI_DMA_ENABLE | RW | DMA enable.<br>1: DMA enabled; 0: DMA disabled. | 0 |

*Note: If the DMA address loop is enabled, when the DMA address is added to the set end address, it will automatically loop to the set first address, without resetting the DMA start address register (R16_SPIx_DMA_BEG) and DMA end address register (R16_SPIx_DMA_END).*

SPI interrupt enable register (R8_SPIx_INTER_EN) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_SPI_IE_FST_BYTE | RW | In the first byte command mode in slave mode, first byte receive interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| [6:5] | Reserved | R0 | Reserved. | 0 |
| 4 | RB_SPI_IE_FIFO_OV | RW | FIFO overflow (FIFO is full when receiving or FIFO is empty when transmitting) interrupt enable:<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 3 | RB_SPI_IE_DMA_END | RW | DMA end interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 2 | RB_SPI_IE_FIFO_HF | RW | FIFO half use interrupt enable:<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 1 | RB_SPI_IE_BYTE_END | RW | SPI single byte transfer complete interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 0 | RB_SPI_IE_CNT_END | RW | SPI all bytes transfer complete interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |

SPI master mode clock divider register (R8_SPIx_CLOCK_DIV) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_SPI_CLOCK_DIV | RW | Division factor in master mode. The minimum value is 2.<br>SPI clock frequency, $F_{SPI}$ = Fsys / division factor. | 10h |

SPI1 slave mode preset data register (R8_SPIx_SLAVE_PRE)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_SPIx_SLAVE_PRE | RW | First return data preset in slave mode.<br>Used to receive the return data after first byte of data. | 10h |

SPI data buffer (R8_SPIx_BUFFER) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_SPIx_BUFFER | RW | SPI data transmit and receive buffer | X |

SPI running flag register (R8_SPIx_RUN_FLAG) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_SPI_SLV_SELECT | RO | Chip select status in slave mode.<br>1: Being selected; 0: Not selected. | 0 |
| 6 | RB_SPI_SLV_CS_LOAD | RO | First load status after chip select in slave mode.<br>1: Being loading R8_SPIx_SLAVE_PRE;<br>0: Not yet loaded or has completed (preload value can be modified). | 0 |
| 5 | RB_SPI_FIFO_READY | RO | FIFO ready status.<br>1: FIFO ready (R16_SPIx_TOTAL_CNT is not 0, and the FIFO is not full when receiving or the FIFO is not empty when transmitting)<br>0: FIFO not ready. | 0 |
| 4 | RB_SPI_SLV_CMD_ACT | RO | Command receive complete status bit in slave mode (the exchange of the first byte of data is completed).<br>1: The first byte has just been exchanged;<br>0: The first byte has not been exchanged or it is not the first byte. | 0 |
| [3:0] | Reserved | R0 | Reserved. | 0 |

SPI interrupt flag register (R8_SPIx_INT_FLAG) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_SPI_IF_FST_BYTE | RW1 | First byte receive flag in slave mode. Cleared by writing 1:<br>1: Received; 0: Not received. | 0 |
| 6 | RB_SPI_FREE | RO | Current SPI idle state.<br>1: Idle; 0: Not idle. | 1 |
| 5 | Reserved | RO | Reserved. | 0 |
| 4 | RB_SPI_IF_FIFO_OV | RW1 | FIFO overflow (FIFO is full when receiving or FIFO is empty when transmitting) flag bit. Cleared by writing 1.<br>1: Overflowed; 0: Not overflowed. | 0 |

| | | | | |
|---|---|---|---|---|
| 3 | RB_SPI_IF_DMA_END | RW1 | DMA complete flag. Cleared by writing 1,<br>1: Completed; 0: Not completed. | 0 |
| 2 | RB_SPI_IF_FIFO_HF | RW1 | FIFO half use (FIFO>=4 when receiving or FIFO<4 when transmitting) flag. Cleared by writing 1:<br>1: FIFO has been half used;<br>0: FIFO has not been half used. | 0 |
| 1 | RB_SPI_IF_BYTE_END | RW1 | SPI single byte transfer complete flag bit. Cleared by writing 1.<br>1: Single byte transfer completed;<br>0: Transfer not completed. | 0 |
| 0 | RB_SPI_IF_CNT_END | RW1 | SPI all bytes transfer complete flag. Cleared by writing 1.<br>1: All transfer completed;<br>0: Transfer not completed. | 0 |

SPI transceiver FIFO count register (R8_SPIx_FIFO_COUNT) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_SPIx_FIFO_COUNT | RW | The byte count in the current FIFO. | 0 |

SPI transceiver FIFO count register (R8_SPIx_FIFO_COUNT1) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_SPIx_FIFO_COUNT1 | RW | The byte count in the current FIFO, which is equivalent to the R8_SPIx_FIFO_COUNT register. | 0 |

SPI transceiver data total length register (R16_SPIx_TOTAL_CNT) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | R16_SPIx_TOTAL_CNT | RW | The total number of bytes of SPI data transmitted and received in master mode, and the lower 12 bits are valid. At most 4095 bytes can be transmitted at a time when using DMA. Slave mode is not supported. | 0 |

SPI FIFO register (R8_SPIx_FIFO) (x=0/1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_SPIx_FIFO | RO/ WO | SPI FIFO register. The FIFO size is 8 bytes. | 0 |

The R8_SPIx_BUFFER register and the R8_SPIx_FIFO register are both SPI data registers, and the main difference between them is:

Reading R8_SPIx_BUFFER is to take the last data exchanged by SPI, which does not affect FIFO or R8_SPIx_FIFO_COUNT. Writing to R8_SPIx_BUFFER in master mode is to transmit the byte directly, and writing operation in slave mode is undefined;

Reading R8_SPIx_FIFO is to take the earliest data exchanged in FIFO, which reduces FIFO and

R8_SPIx_FIFO_COUNT. Writing to R8_SPIx_FIFO is to temporarily store data in FIFO. In slave mode, the external SPI master decides when to take the data. In master mode, automatically startup transmission when R16_SPIx_TOTAL_CNT is not 0.

Current address of SPI DMA buffer (R32_SPIx_DMA_NOW)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:18] | Reserved | RO | Reserved. | 0 |
| [17:0] | R16_SPIx_DMA_NOW | RW | Current address of DMA data buffer.<br>It can be used to calculate the number of conversions.<br>CNT=R16_SPIx_DMA_NOW-R16_SPIx_DMA_BEG | X |

Start address of SPI DMA buffer (R32_SPIx_DMA_BEG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:18] | Reserved | RO | Reserved. | 0 |
| [17:0] | R16_SPIx_DMA_BEG | RW | Start address of DMA data buffer. 4 bytes must be aligned for the address. | X |

*Note: This DMA address can cover RAMS and RAMX areas.*

End address of SPI DMA buffer (R32_SPIx_DMA_END)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:18] | Reserved | RO | Reserved. | 0 |
| [17:0] | R16_SPIx_DMA_END | RW | End address of DMA data buffer (not included). 4 bytes must be aligned for the address. | X |

*Note: This DMA address can cover RAMS and RAMX areas.*

# Chapter 7 Universal Asynchronous Receiver Transmitter (UART)

The system is equipped with 4 full-duplex UARTs (UART0/1/2/3). Both full-duplex and half-duplex communication are supported. UART0 is equipped with the transmit status pin for switching RS485, and it supports MODEM signals CTS, DSR, RI, DCD, DTR and RTS.

## 7.1 Main features

- Compatible with 16C550 asynchronous serial port, and enhanced.
- 5/6/7/8 data bits, 1/2 stop bits
- Supports odd parity, even parity, no parity, space 0 parity and mark 1 parity
- Programmable baud rate. Support 115200bps baud rate and up to 6Mbps
- Built-in 8-byte FIFO buffer. Supports 4 FIFO trigger levels
- UART0 supports MODEM signals: CTS, DSR, RI, DCD, DTR and RTS, which can be converted to RS232 level
- Supports automatic handshake and automatic transfer rate control of hardware flow control signals CTS and RTS, compatible with TL16C550C
- Supports serial port frame error detection and Break line interval detection
- Full-duplex and half-duplex communication are supported, and UART0 provides a transmit status pin for switching RS485

## 7.2 Functional description

UART0/1/2/3 output pins are all 3.3V TTL levels The pins in asynchronous serial port mode include: data transmission pin (supported by UART0/1/2/3) and MODEM contact signal pin (only supported by UART0). Data transmission pins include: TXD pin and RXD pin, both of which are at high level by default. MODEM contact signal pins include: CTS pin, DSR pin, RI pin, DCD pin, DTR pin and RTS pin, all of which are at high level by default. All these MODEM contact signal pins can be used as general-purpose I/O pins, controlled by the application program and their purposes can be defined.

The 4 UARTs each has built-in independent receive/transmit buffer and 8-byte FIFO, and they support simplex, half-duplex and full-duplex asynchronous serial communication. Serial data includes 1 low-level start bit, 5/6/7/8 data bits, 0/1 additional parity/flag bit, and 1/2 high-level stop bits, and supports odd/even/mark/space parity.

The module supports a baud rate of up to 7.5Mbps. For commonly used baud rates, it is necessary to select an appropriate module reference clock and division factor to reduce the calculation errors of hardware. The baud rate error of UART transmit signal is less than 0.2%, and the allowable baud rate error of UART receive signal is not greater than 2%.

### 7.2.1 Baud rate calculation

1) Calculate the reference clock. Set the R8_UART0_DIV register, with the maximum value of 127;
2) Calculate the baud rate. Set the R16_UART0_DL register;

   Baud rate = Fsys * 2 / R8_UART0_DIV / 16 / R16_UART0_DL.

### 7.2.2 UART transmission

"THR register empty" interrupt sent by UART (the lower 4 bits of IIR are 02H) refers to transmit FIFO empty. The interrupt is cleared when the IIR is read, or the interrupt can also be cleared when the next data is written to THR. If only one byte is written to THR, it will soon generate a request of "Transmit Hold Register (THR) empty interrupt" as the byte is quickly transferred to the Transmit Shift Register (TSR) to start transmission. In this case, the next data ready to be transmitted can be written. After all data in TSR is removed, UART transmission is completed, and the RB_LSR_TX_ALL_EMP bit in LSR becomes 1 and valid.

In interrupt trigger mode, when THR empty interrupt is received from UART, if FIFO is enabled, up to 8 bytes can be written to THR and FIFO at a time and will be transmitted automatically by the controller in sequence. If FIFO is disabled, only one byte can be written at a time. If no data needs to be transmitted, directly exit (the interrupt has been automatically cleared when IIR is read earlier).

In the query mode, it can judge whether the transmit FIFO is empty according to RB_LSR_TX_FIFO_EMP bit in LSR. If it is 1, data can be written to THR and FIFO. If FIFO is enabled, up to 8 bytes can be written at a time.

### 7.2.3 UART reception

UART receive data available interrupt (the lower 4 bits in IIR are 04H) means that the number of data bytes in the receive FIFO has reached or exceeded the FIFO trigger point set and selected by RB_FCR_FIFO_TRIG in the FCR register. The interrupt is cleared when the data is read from RBR to cause the number of bytes in the FIFO lower than the FIFO trigger point.

UART receive data timeout interrupt (the lower 4 bits in IIR are 0CH) means that there is at least one byte of data in the receive FIFO, and it has waited for the time equivalent to receiving 4 data bytes when UART receives data last time and the MCU takes the data last time. The interrupt is cleared when a new data is received again, or the interrupt can also be cleared when the MCU reads RBR once. When the receive FIFO is empty, the RB_LSR_DATA_RDY bit in LSR is 0. When there is data in the receive FIFO, the RB_LSR_DATA_RDY bit becomes 1 and valid.

In the interrupt trigger mode, when the UART receive data timeout interrupt is received, the R8_UARTx_RFC register can be read to query the remaining data count in the current FIFO, and all data can be read directly. Or continuously query the RB_LSR_DATA_RDY bit in LSR. If this bit is valid, data can be read until this bit becomes invalid. After the UART receive data available interrupt is received, you can read the number of bytes set by RB_FCR_FIFO_TRIG in FCR from RBR, and then directly read the data for the number of bytes, or you can read all the data in the current FIFO according to the RB_LSR_DATA_RDY bit and the R8_UARTx_RFC register.

In query mode, the MCU can judge whether the receive FIFO is empty according to the RB_LSR_DATA_RDY bit in LSR, or read the R8_UARTx_RFC register to get the data count in the current FIFO and get all the data received by UART.

### 7.2.4 Hardware flow control

Hardware flow control includes auto CTS (RB_MCR_AU_FLOW_EN in MCR is 1) and auto RTS (RB_MCR_AU_FLOW_EN and RB_MCR_RTS in MCR are 1).

If auto CTS is enabled, CTS pin must be valid before UART transmits data. The UART transmitter detects CTS pin before the next data transmission. When the CTS pin is valid, the transmitter sends the next data. In order to ensure that the transmitter stops transmitting the later data, the CTS pin must be disabled before the middle of the last stop bit currently transmitted. The auto CTS function reduces the interrupt applied to the MCU system. When hardware flow control is enabled, a change in CTS pin level does not trigger a MODEM interrupt as the controller automatically controls the transmitter based on CTS pin state. If auto RTS is enabled, RTS pin output is valid only when there is enough space in FIFO to receive data, and RTS pin output is disabled when the receive FIFO is full. RTS pin output is valid if all data in the receive FIFO is taken or cleared. When the trigger point for the receive FIFO is reached (the number of existing bytes in the receive FIFO is not less than the number of bytes set by RB_FCR_FIFO_TRIG in FCR), RTS pin output is invalid, and the transmitter of the other side is allowed to transmit another data after RTS pin is invalid. Once the data in the receive FIFO is emptied, RTS pin is automatically re-enabled, so that the transmitter of the other side resumes transmitting. If both auto CTS and auto RTS are enabled (both RB_MCR_AU_FLOW_EN and RB_MCR_RTS in MCR are 1), one side will not transmit data unless there is sufficient space in the receive FIFO of the other side when RTS pin of one side is connected to CTS pin of the other side. Therefore, the hardware flow control can avoid FIFO overflow and timeout errors for UART reception.

## 7.3 Register description

UART0 register physical base address: 0x40003000
UART1 register physical base address: 0x40003400
UART2 register physical base address: 0x40003800
UART3 register physical base address: 0x40003C00

Table 7-1 UART0 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_UART0_MCR | 0x00 | MODEM control register | 0x00 |
| R8_UART0_IER | 0x01 | Interrupt enable register | 0x00 |
| R8_UART0_FCR | 0x02 | FIFO control register | 0x00 |
| R8_UART0_LCR | 0x03 | Line control register | 0x00 |
| R8_UART0_IIR | 0x04 | Interrupt identification register | 0x01 |
| R8_UART0_LSR | 0x05 | Line status register | 0xC0 |
| R8_UART0_MSR | 0x06 | MODEM status register | 0xX0 |
| R8_UART0_RBR | 0x08 | Receive buffer register | 0xXX |
| R8_UART0_THR | 0x08 | Transmit hold register | 0xXX |
| R8_UART0_RFC | 0x0A | Receive FIFO count register | 0xXX |
| R8_UART0_TFC | 0x0B | Transmit FIFO count register | 0xXX |
| R16_UART0_DL | 0x0C | Baud rate divisor latch | 0xXXXX |
| R8_UART0_DIV | 0x0E | Prescaler divisor register | 0xXX |
| R8_UART0_ADR | 0x0F | Slave address register | 0xFF |

Table 7-2 UART1 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_UART1_MCR | 0x00 | MODEM control register | 0x00 |

| | | | |
|---|---|---|---|
| R8_UART1_IER | 0x01 | Interrupt enable register | 0x00 |
| R8_UART1_FCR | 0x02 | FIFO control register | 0x00 |
| R8_UART1_LCR | 0x03 | Line control register | 0x00 |
| R8_UART1_IIR | 0x04 | Interrupt identification register | 0x01 |
| R8_UART1_LSR | 0x05 | Line status register | 0xC0 |
| R8_UART1_RBR | 0x08 | Receive buffer register | 0xXX |
| R8_UART1_THR | 0x08 | Transmit hold register | 0xXX |
| R8_UART1_RFC | 0x0A | Receive FIFO count register | 0xXX |
| R8_UART1_TFC | 0x0B | Transmit FIFO count register | 0xXX |
| R16_UART1_DL | 0x0C | Baud rate divisor latch | 0xXXXX |
| R8_UART1_DIV | 0x0E | Prescaler divisor register | 0xXX |

Table 7-3 UART2 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_UART2_MCR | 0x00 | MODEM control register | 0x00 |
| R8_UART2_IER | 0x01 | Interrupt enable register | 0x00 |
| R8_UART2_FCR | 0x02 | FIFO control register | 0x00 |
| R8_UART2_LCR | 0x03 | Line control register | 0x00 |
| R8_UART2_IIR | 0x04 | Interrupt identification register | 0x01 |
| R8_UART2_LSR | 0x05 | Line status register | 0xC0 |
| R8_UART2_RBR | 0x08 | Receive buffer register | 0xXX |
| R8_UART2_THR | 0x08 | Transmit hold register | 0xXX |
| R8_UART2_RFC | 0x0A | Receive FIFO count register | 0xXX |
| R8_UART2_TFC | 0x0B | Transmit FIFO count register | 0xXX |
| R16_UART2_DL | 0x0C | Baud rate divisor latch | 0xXXXX |
| R8_UART2_DIV | 0x0E | Prescaler divisor register | 0xXX |

Table 7-4 UART3 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_UART3_MCR | 0x00 | MODEM control register | 0x00 |
| R8_UART3_IER | 0x01 | Interrupt enable register | 0x00 |
| R8_UART3_FCR | 0x02 | FIFO control register | 0x00 |
| R8_UART3_LCR | 0x03 | Line control register | 0x00 |
| R8_UART3_IIR | 0x04 | Interrupt identification register | 0x01 |
| R8_UART3_LSR | 0x05 | Line status register | 0xC0 |
| R8_UART3_RBR | 0x08 | Receive buffer register | 0xXX |
| R8_UART3_THR | 0x08 | Transmit hold register | 0xXX |
| R8_UART3_RFC | 0x0A | Receive FIFO count register | 0xXX |
| R8_UART3_TFC | 0x0B | Transmit FIFO count register | 0xXX |
| R16_UART3_DL | 0x0C | Baud rate divisor latch | 0xXXXX |
| R8_UART3_DIV | 0x0E | Prescaler divisor register | 0xXX |

Modulator-Demodulator (MODEM) control register (R8_UARTx_MCR) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_MCR_HALF | RW | Half-duplex receive/transmit mode enable (only supported by UART0).<br>1: Half-duplex receive/transmit mode enabled. Transmission has priority. Receive when not transmit;<br>0: Half-duplex receive/transmit mode disabled. | 0 |
| 6 | RB_MCR_TNOW | RW | DTR pin output transmit now status (TNOW) enable (only supported by UART0), can be used to control RS485 receive/transmit switch.<br>1: Enabled; 0: Disabled. | 0 |
| 5 | RB_MCR_AU_FLOW_EN | RW | CTS and RTS hardware automatic flow enable<br>1: Enabled; 0: Disabled.<br>In the flow control mode, if this bit is 1, UART continues to transmit the next data only when it detects that the CTS pin input is valid (active low). Otherwise, UART transmission is paused, and the CTS input status change will not generate the MODEM status interrupt when this bit is 1. If this bit is 1 and RTS is 1, UART automatically validates the RTS pin (active low) when the receive FIFO is empty. UART automatically invalidates the RTS pin when the number of received bytes reaches the trigger point of FIFO and UART re-validates the RTS pin when the receive FIFO is empty. You can connect your own CTS pin to the other party's RTS pin through hardware auto band rate control, and connect your own RTS pin to the other party's CTS pin. | 0 |
| 4 | RB_MCR_LOOP | RW | Test mode enable of internal loop (only supported by UART0).<br>1: Enabled; 0: Disabled.<br>In the test mode of the internal loop, all external output pins of UART are invalid, TXD internally returns to RXD (i.e., the output of TSR internally returns to the input of RSR), RTS internally returns to CTS, DTR internally returns to DSR, OUT1 internally returns to RI and OUT2 internally returns to DCD. | 0 |
| 3 | RB_MCR_OUT2 | RW | UART interrupt request enable control.<br>1: Enabled; 0: Disabled. | 0 |
| 2 | RB_MCR_OUT1 | RW | User-defined MODEM control bit (only supported by UART0), and no actual output pin is connected:<br>1: Set high; 0: Set low. | 0 |
| 1 | RB_MCR_RTS | RW | RTS pin output level control (only supported by UART0).<br>1: RTS signal output is valid (low level);<br>0: RTS signal output high level (default). | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | RB_MCR_DTR | RW | DTR pin output level control (only supported by UART0).<br>1: DTR signal output is valid (low level);<br>0: DTR signal output high level (default). | 0 |

Interrupt enable register (R8_UARTx_IER) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_IER_RESET | WZ | UART software reset control. Automatically cleared by hardware.<br>1: Reset UART; 0: Normal operation. | 0 |
| 6 | RB_IER_TXD_EN | RW | UART TXD pin output enable.<br>1: Output enabled; 0: Output disabled. | 0 |
| 5 | RB_IER_RTS_EN | RW | RTS pin output enable (only supported by UART0).<br>1: Output enabled; 0: Output disabled. | 0 |
| 4 | RB_IER_DTR_EN | RW | DTR pin output enable (only supported by UART0).<br>1: Output enabled; 0: Output disabled. | 0 |
| 3 | RB_IER_MODEM_CHG | RW | UART0 Modem input status change interrupt enable (only supported by UART0).<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 2 | RB_IER_LINE_STAT | RW | Receive line status interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 1 | RB_IER_THR_EMPTY | RW | Transmit hold register empty interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 0 | RB_IER_RECV_RDY | RW | Receive data interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |

FIFO control register (R8_UARTx_FCR) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:6] | RB_FCR_FIFO_TRIG | RW | Interrupt of the receive FIFO and trigger point of hardware flow control setting domain.<br>00: 1 byte; 01: 2 bytes;<br>10: 4 bytes; 11: 7 bytes.<br>This domain is used to set the interrupt of the receive FIFO and trigger point of hardware flow control. For example: 00 corresponds to 1 byte, that is, the receive data available interrupt is generated when 1 byte is received, and RTS pin is automatically disabled when hardware flow control is enabled. | 0 |
| [5:3] | Reserved | RO | Reserved | 0 |
| 2 | RB_FCR_TX_FIFO_CLR | WZ | Transmit FIFO data clear enable. Automatically cleared by hardware.<br>1: Data in the transmit FIFO cleared (excluding TSR);<br>0: Data in the transmit FIFO not cleared. | 0 |

| | | | | |
|---|---|---|---|---|
| 1 | RB_FCR_RX_FIFO_CLR | WZ | Receive FIFO data clear enable. Automatically cleared by hardware.<br>1: Data in the receive FIFO cleared (excluding RSR);<br>0: Data in the receive FIFO not cleared. | 0 |
| 0 | RB_FCR_FIFO_EN | RW | FIFO enable.<br>1: FIFO enabled. The size of the internal FIFO is 8 bytes;<br>0: FIFO disabled.<br>After FIFO is disabled, it is 16C450 compatible mode, which means that there is only one byte in FIFO (RECV_TG1=0, RECV_TG0=0, FIFO_EN=1). It is recommended to enable FIFO. | 0 |

Line control register (R8_UARTx_LCR) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_LCR_DLAB/<br>RB_LCR_GP_BIT | RW | UART general purpose bit, user-defined. | 0 |
| 6 | RB_LCR_BREAK_EN | RW | Forced to generate BREAK line interval enable.<br>1: Forced to generate; 0: Not generated. | 0 |
| [5:4] | RB_LCR_PAR_MOD | RW | Parity bit format setting. Valid only when the RB_LCR_PAR_EN bit is 1.<br>00: Odd parity;<br>01: Even parity;<br>10: Mark (set to 1);<br>11: Space (cleared). | 0 |
| 3 | RB_LCR_PAR_EN | RW | Parity bit enable.<br>1: Parity bit generation enabled during transmission and parity bit check enabled during reception;<br>0: No parity bit. | 0 |
| 2 | RB_LCR_STOP_BIT | RW | Stop bit format setting.<br>1: 2 stop bits; 0: 1 stop bit. | 0 |
| [1:0] | RB_LCR_WORD_SZ | RW | UART word size setting.<br>00: 5 data bits; 01: 6 data bits;<br>10: 7 data bits; 11: 8 data bits. | 0 |

Interrupt identification register (R8_UARTx_IIR) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:6] | RB_IIR_FIFO_ID | RO | UART FIFO enable.<br>11: FIFO enabled; 00: FIFO not enabled. | 0 |
| [5:4] | Reserved | R0 | Reserved. | 0 |
| [3:0] | RB_IIR_INT_MASK | RO | Interrupt flag domain: If the RB_IIR_NO_INT bit is 0, an interrupt is generated, and it needs to read this domain to determine the interrupt source. Please refer to Table 7-3 for details. | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | RB_IIR_NO_INT | RO | UART no interrupt flag.<br>1: No interrupt; 0: Interrupt. | 1 |

Table 7-3 Meaning of RB_IIR_INT_MASK in IIR register

| IIR register bit | | | | Priority | Interrupt Type | Interrupt source | Clear interrupt |
|---|---|---|---|---|---|---|---|
| IID3 | IID2 | IID1 | NOINT | | | | |
| 0 | 0 | 0 | 1 | None | No interrupt generated | No interrupt | |
| 1 | 1 | 1 | 0 | 0 | Bus address matching | The received 1 data is the UART bus address, and the address matches the preset slave address or the broadcast address. Note: This interrupt only applies to UART0. | Read IIR or disable multi-device mode |
| 0 | 1 | 1 | 0 | 1 | Receive line status | OVER_ERR, PAR_ERR, FRAM_ERR, BREAK_ERR | Read LSR |
| 0 | 1 | 0 | 0 | 2 | Receive data available | The number of bytes received reaches the trigger point of FIFO. | Read RBR |
| 1 | 1 | 0 | 0 | 2 | Receive data timeout | No next data is received after more than 4 data times. | Read RBR |
| 0 | 0 | 1 | 0 | 3 | THR empty | Transmit hold register empty, or RB_IER_THR_EMPTY bit is changed from 0 to 1 for triggering. | Read IIR or write to THR |
| 0 | 0 | 0 | 0 | 4 | MODEM input change | Trigger by setting △ CTS, △ DSR, △ RI and △ DCD to 1. | Read MSR |

Line status register (R8_UARTx_LSR) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_LSR_ERR_RX_FIFO | RO | Receive FIFO error flag bit:<br>1: There is at least one PAR_ERR, FRAM_ERR or BREAK_ERR error in the receive FIFO;<br>0: No error in the receive FIFO. | 0 |
| 6 | RB_LSR_TX_ALL_EMP | RO | Transmit hold register (THR) and transmit shift register (TSR) empty flag.<br>1: Both are empty; 0: Not both are empty. | 1 |
| 5 | RB_LSR_TX_FIFO_EMP | RO | Transmit FIFO empty flag.<br>1: Transmit FIFO empty;<br>0: Transmit FIFO not empty. | 1 |
| 4 | RB_LSR_BREAK_ERR | RZ | BREAK line interval detection flag.<br>1: BREAK line interval detected;<br>0: BREAK line interval not detected. | 0 |
| 3 | RB_LSR_FRAME_ERR | RZ | Data frame error flag.<br>1: There is frame error of the data being read from the receive FIFO due to lack of valid stop bit.<br>0: The currently read data frame is correct. | 0 |
| 2 | RB_LSR_PAR_ERR | RZ | Parity error flag of received data. | 0 |

| | | | | |
|---|---|---|---|---|
| | | | 1: There is parity error of the data being read from the receive FIFO;<br>0: The currently read data parity is correct. | |
| 1 | RB_LSR_OVER_ERR | RZ | Receive FIFO buffer overflow flag.<br>1: Overflowed; 0: Not overflowed. | 0 |
| 0 | RB_LSR_DATA_RDY | RO | Data flag in the receive FIFO.<br>1: There is data in FIFO; 0: No data. | 0 |

Modulator-demodulator (MODEM) status register (R8_UART0_MSR) (only supported by UART0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_MSR_DCD | RO | DCD pin status.<br>1: DCD pin is valid (low level);<br>0: DCD pin is invalid (high level). | x |
| 6 | RB_MSR_RI | RO | RI pin status.<br>1: RI pin is valid (low level);<br>0: RI pin is invalid (high level). | x |
| 5 | RB_MSR_DSR | RO | DSR pin status.<br>1: DSR pin is valid (low level);<br>0: DSR pin is invalid (high level). | x |
| 4 | RB_MSR_CTS | RO | CTS pin status.<br>1: CTS pin is valid (low level);<br>0: CTS pin is invalid (high level). | x |
| 3 | RB_MSR_DCD_CHG | RZ | DCD pin input status change flag.<br>1: Changed; 0: Not changed. | 0 |
| 2 | RB_MSR_RI_CHG | RZ | RI pin input status change flag.<br>1: Changed; 0: Not changed. | 0 |
| 1 | RB_MSR_DSR_CHG | RZ | DSR pin input status change flag.<br>1: Changed; 0: Not changed. | 0 |
| 0 | RB_MSR_CTS_CHG | RZ | CTS pin input status change flag.<br>1: Changed; 0: Not changed. | 0 |

Receive buffer register (R8_UARTx_RBR) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_UARTx_RBR | RO | Data receive buffer register.<br>If the DATA_RDY bit in LSR is 1, the received data can be read from this register;<br>If FIFO_EN is 1, the data received from the UART shift register (RSR) is firstly stored in the receive FIFO, and then read out through the register. | X |

Transmit hold register (R8_UARTx_THR) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_UARTx_THR | WO | Transmit hold register. | X |

| | | | | |
|---|---|---|---|---|
| | | | Including the transmit FIFO, for writing the data to be transmitted. If FIFO_EN is 1, the written data is firstly stored in the transmit FIFO, and then output one by one through the transmit shift register (TSR). | |

Receive FIFO count register (R8_UARTx_RFC) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_UARTx_RFC | RO | Data count in the current receive FIFO. The maximum value is 8. | X |

Transmit FIFO count register (R8_UARTx_TFC) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_UARTx_TFC | RO | Data count in the current transmitter FIFO. The maximum value is 8. | X |

Baud rate divisor latch (R16_UARTx_DL) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | R16_UARTx_DL | RW | 16-bit divisor, which is used to calculate the baud rate.<br>Formula: Divisor = UART internal reference clock ($F_{uart}$) / 16 / baud rate.<br>For example: If UART internal reference clock is 1.8432MHz and the required baud rate is 9600bps, then the divisor =1843200/16/9600=12. | X |

Prescaler divisor register (R8_UARTx_DIV) (x=0/1/2/3)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_UARTx_DIV | RW | Used to calculate UART internal reference clock.<br>Formula: Divisor = Fsys*2 / UART internal reference clock, with a maximum value of 127. | X |

Slave address register (R8_UART0_ADR), only used by UART0

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_UART0_ADR | RW | UART0 slave address.<br>FFh: Not used; Others: Slave address. | FFh |

R8_UART0_ADR presets the address of this device when it acts as a slave, which is used to automatically compare the received addresses during multi-device communication, and generate an interrupt when the address matches or when the broadcast address 0FFH is received. Meanwhile, it is allowed to receive subsequent data packets. It is not allowed to receive any data before the address does not match. After sending data or rewriting to the R8_UART0_ADR register, it stops receiving any data, until the address is

matched again next time or until the broadcast address is received.

When R8_UART0_ADR is 0FFH or when RB_LCR_PAR_EN=0, the automatic comparison function of bus address is disabled.

When R8_UART0_ADR is not 0FFH and RB_LCR_PAR_EN=1, the automatic comparison function of bus address is enabled, and the following parameters should be configured: Set the RB_LCR_WORD_SZ control bits to 11b to select 8 data bits. When the address byte is MARK (that is, the bit9 of data byte is 0), RB_LCR_PAR_MOD should be set to 10b. When the address byte is SPACE (that is, the bit9 of data byte is 1), RB_LCR_PAR_MOD should be set to 11b.

# Chapter 8 General purpose timer (TMRx)

The system is equipped with 3 26-bit timers (TMR0, TMR1 and TMR2) and the longest timing interval is 2^26 clock cycles. All timers support capture, PWM and interrupt functions. In addition, TMR1 and TMR2 support DMA function.

## 8.1 Main features

- 3 26-bit timers. The longest timing interval of each timer is 2^26 clock cycles;
- Each timer supports PWM function;
- Each timer supports capture function;
- Each timer supports timer interrupt. In addition, TMR1 and TMR2 support DMA and interrupt;
- The capture function can be set to level change capture function and high/low level hold time capture function;
- PWM function supports to dynamically adjust PWM duty cycle settings;

## 8.2 Functional description

### 8.2.1 Timing and counting function

The longest timing interval supported by these 3 timers is 2^26 clock cycles. If the system clock cycle is 96M, the longest time interval is: 10.4ns*2^26 ≈ 0.7s. If the system clock is lower than 96M, the timing interval is longer.

The timing function is set as follows:

1. Set the R32_TMRx_CNT_END register to the time value that is needed;
   Specific calculation: Time = Fsys * R32_TMRx_CNT_END
2. Set the RB_TMR_MODE_IN bit in the R8_TMRx_CTRL_MOD register to 0, and set the RB_TMR_ALL_CLEAR bit to 0;
3. Set the RB_TMR_COUNT_EN bit in the R8_TMRx_CTRL_MOD register to 1, to start timer function;
4. At the end of timing interval, set the RB_TMR_IF_CYC_END bit in the R8_TMRx_INT_FLAG register to 1, and cleared by writing 1. If the corresponding interrupt bit is enabled, the interrupt service will be triggered.

### 8.3.2 PWM function

Each timer is equipped with PWM output function. The default output polarity of PWM can be set to high level or low level. The number of repeated times can be selected as 1, 4, 8 or 16. This repeat function is combined with DMA to imitate the effect of DAC. The shortest time cycle for PWM output is 1 system clock cycle, and the duty cycle of PWM can be dynamically modified to imitate special waveforms, such as quasi-sine-wave.

It is necessary to set the R32_TMRx_FIFO register as the data register and set the R32_TMRx_CNT_END register as the PWM total cycle register when PWM outputs,.

PWM function is set as follows:

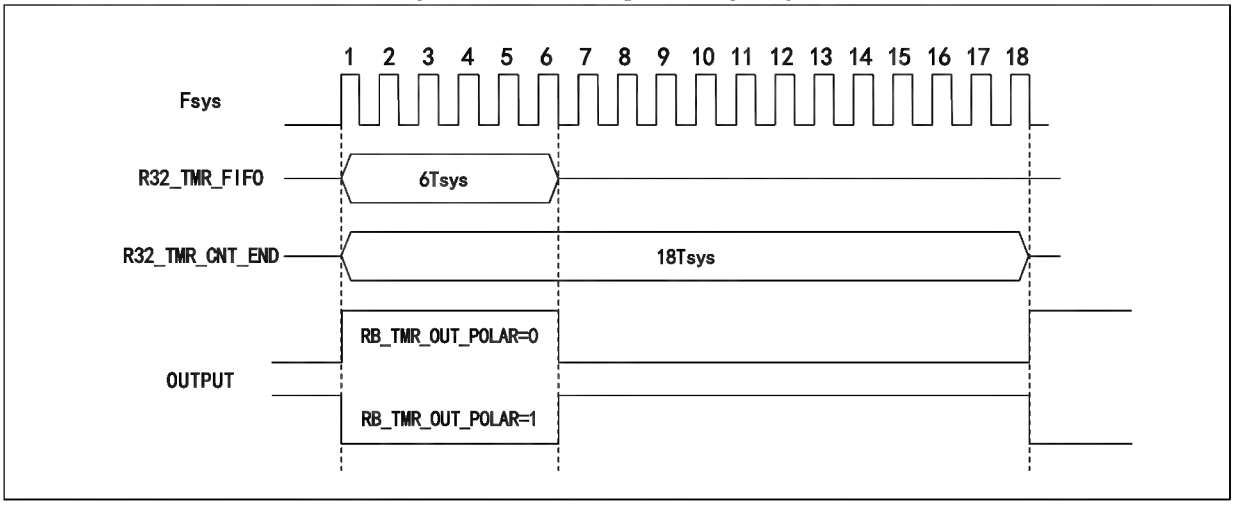
1. Set the PWM total cycle register (R32_TMRx_CNT_END), the minimum value is 1, the value of this register must be greater than or equal to the value of the R32_TMRx_FIFO register;
2. Set the R32_TMRx_FIFO data register, the minimum value is 0, with the corresponding duty cycle of 0%;

the maximum value is the same as that of R32_TMR_CNT_END, with the corresponding duty cycle of 100%. Calculation of duty cycle: R32_TMRx_FIFO/R32_TMRx_CNT_END. TMR1 and TMR2 support continuous dynamic data (DMA), which can imitate special waveforms;

3. Clear the RB_TMR_MODE_IN bit in the mode setting register (R8_TMRx_CTRL_MOD) to 0 to enable the PWM mode. At the same time, set the RB_TMR_ALL_CLEAR bit to 1 and then clear it to forcefully clear the FIFO and COUNT. Set the RB_TMR_OUT_POLAR bit to select the output polarity. If you need to set the number of repetitions, set RB_TMR_PWM_REPEAT as needed.
4. Set the RB_TMR_COUNT_EN bit and the RB_TMR_OUT_EN bit in the mode setting register (R8_TMRx_CTRL_MOD) to 1 to enable the PWM function;
5. Set the I/O pin corresponding to PWM as output;
6. If you need to enable interrupts, set the corresponding interrupt enable register bit;
7. After the PWM is completed, if the interrupt is enabled, the corresponding timer interrupt is generated. In this case, read the R8_TMRx_INT_FLAG register to know whether the PWM is completed and whether an error occurred during the PWM process;

For example: Set the RB_TMR_OUT_POLAR bit to 0, set R32_TMRx_FIFO to 6, set R32_TMRx_CNT_END to 18, and the basic timing diagram of PWM generation is as follows, and its duty cycle is: PWM duty cycle = R32_TMRx_FIFO/R32_TMRx_CNT_END = 1/3

Figure 8-1 PWM output timing diagram




If RB_TMR_PWM_REPEAT is set to 00, it means that the above process is repeated once
01 means that the above process is repeated for 4 times
10 means that the above process is repeated for 8 times
11 means that the above process is repeated for 16 times.

After repeating, take the next data in FIFO and then continue.

### 8.3.3 Capture function

Each timer has a capture function, among which the capture functions of TMR1 and TMR2 support DMA data storage. Capture modes can be selected: start triggering at any edge and end at any edge, start triggering at rising edge and end at rising edge, and start triggering at falling edge and end at falling edge. The following table shows the description of capture trigger mode:

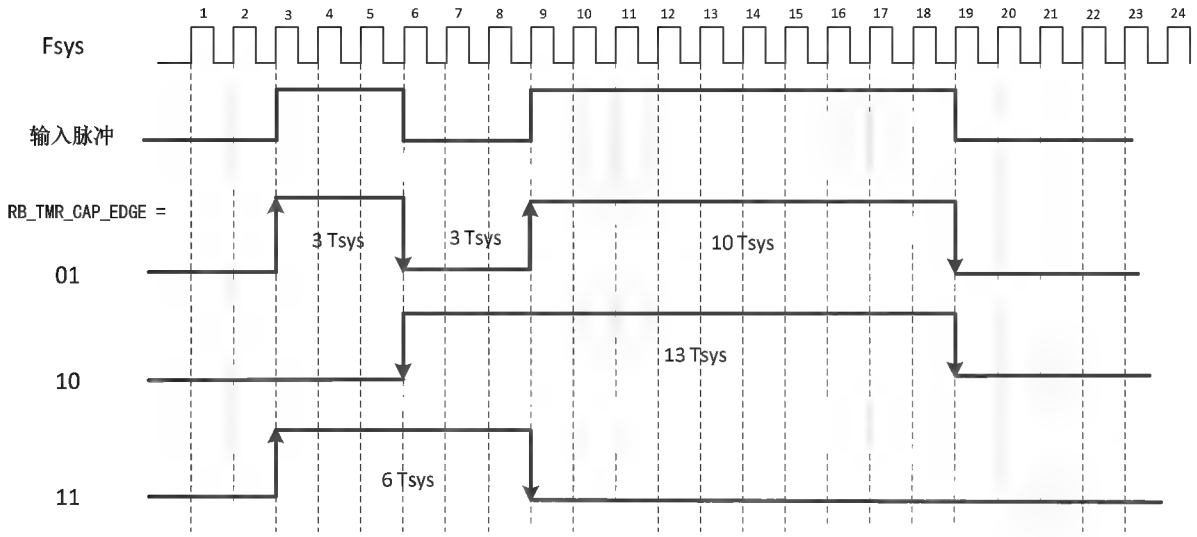
Table 8-1 Description of capture trigger mode

| Capture mode selection bit (RB_TMR_CAP_EDGE) | Trigger bode | Icon |
|---|---|---|
| 00 | Capture disabled | None |
| 01 | Edge trigger | |
| 10 | Falling edge to falling edge | |
| 11 | Rising edge to rising edge | |

There are 2 trigger states in edge trigger mode, which can capture high level width or low level width. When highest bit (bit25) of the valid data in data register (R32_TMRx_FIFO) is 1, high level is captured. Otherwise, low level is captured. If the bit25 of multiple sets of data is 1 (or 0), the width of the high (or low) level exceeds the timeout value, and accumulation of multiple sets are required.

In the trigger modes from falling edge to falling edge or from rising edge to rising edge, an input change cycle can be captured. When the highest bit (bit25) of the valid data in data register (R32_TMRx_FIFO) is 0, one cycle is normally sampled. When it is 1, the input change cycle exceeds the timeout value R32_TMRx_CNT_END, and the latter set of data needs to be added and accumulated as a single input change period. As shown in Figure 8-2:

Figure 8-2 Taking system clock cycle as capture cycle




When RB_TMR_CAP_EDGE = 01b, it is set as edge-triggered sampling, and widths of the time sampled are 3Tsys, 3Tsys, 10Tsys;

When RB_TMR_CAP_EDGE = 10b, it is set as the sampling from falling edge to falling edge, and width of the time sampled is 13Tsys;

When RB_TMR_CAP_EDGE = 11b, it is set as the sampling from rising edge to rising edge, and width of the time sampled is 6Tsys;

The capture function is set as follows:

1. Set the R32_TMRx_CNT_END register to set the capture timeout time. The default maximum timeout

time is 2^26 clock cycles. It is recommended to set a reasonable timeout to avoid no data when the input is unchanged for a long time. If no level change is detected within the maximum timeout time, set the bit25 in the R32_TMRx_FIFO register to 1;

2. Set the direction of the I/O pin corresponding to capture as input;
3. Set the RB_TMR_MODE_IN bit in the mode setting register (R8_TMRx_CTRL_MOD) to 1 and set the RB_TMR_CAP_COUNT bit to 0, and reset the RB_TMR_ALL_CLEAR bit to clear FIFO and COUNT. At the same time, set the RB_TMR_CAP_EDGE control domain to select the capture mode;
4. If interrupt needs to be enabled, set the corresponding bit in the interrupt register R8_TMRx_INTER_EN to 1, to enable the corresponding interrupt;
5. To save the captured data by DMA (only supported by TMR1 and TMR2), you need to set the RB_TMR_DMA_ENABLE bit in the R8_TMRx_CTRL_DMA register to 1, to enable DMA, and set the R16_TMRx_DMA_BEG register to the first address of the buffer for storing the captured data, and set the R16_TMRx_DMA_END register as the end address of the buffer for storing captured data;
6. Set the RB_TMR_COUNT_EN bit in the R8_TMRx_CTRL_MOD register to 1, to enable timer module and start the capture function;
7. After the capture is completed, the R8_TMRx_INT_FLAG register will generate the corresponding interrupt status. The data captured by default is stored in the R32_TMRx_FIFO register. If DMA data transmission is used, the captured data is automatically stored in the data buffer set by DMA.

## 8.3 Register description

TMR0 register physical base address: 0x40002000
TMR1 register physical base address: 0x40002400
TMR2 register physical base address: 0x40002800

Table 8-2 TMR0 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_TMR0_CTRL_MOD | 0x00 | Mode setting register | 0x02 |
| R8_TMR0_INTER_EN | 0x02 | Interrupt enable register | 0x00 |
| R8_TMR0_INT_FLAG | 0x06 | Interrupt flag register | 0x00 |
| R8_TMR0_FIFO_COUNT | 0x07 | FIFO count register | 0x00 |
| R32_TMR0_COUNT | 0x08 | Current count value register | 0x00000000 |
| R32_TMR0_CNT_END | 0x0C | Count end value setting register | 0x00000000 |
| R32_TMR0_FIFO | 0x10 | FIFO register | 0x00000000 |

Table 8-3 TMR1 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_TMR1_CTRL_MOD | 0x00 | Mode setting register | 0x02 |
| R8_TMR1_CTRL_DMA | 0x01 | DMA control register | 0x00 |
| R8_TMR1_INTER_EN | 0x02 | Interrupt enable register | 0x00 |
| R8_TMR1_INT_FLAG | 0x06 | Interrupt flag register | 0x00 |
| R8_TMR1_FIFO_COUNT | 0x07 | FIFO count register | 0x00 |

| | | | |
|---|---|---|---|
| R32_TMR1_COUNT | 0x08 | Current count value register | 0x00000000 |
| R32_TMR1_CNT_END | 0x0C | Final count value register | 0x00000000 |
| R32_TMR1_FIFO | 0x10 | FIFO register | 0x00000000 |
| R32_TMR1_DMA_NOW | 0x14 | Current address of DMA buffer | 0x0000 |
| R32_TMR1_DMA_BEG | 0x18 | Start address of DMA buffer | 0x0000 |
| R32_TMR1_DMA_END | 0x1C | End address of DMA buffer | 0x0000 |

Table 8-4 TMR2 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_TMR2_CTRL_MOD | 0x00 | Mode setting register | 0x02 |
| R8_TMR2_CTRL_DMA | 0x01 | DMA control register | 0x00 |
| R8_TMR2_INTER_EN | 0x02 | Interrupt enable register | 0x00 |
| R8_TMR2_INT_FLAG | 0x06 | Interrupt flag register | 0x00 |
| R8_TMR2_FIFO_COUNT | 0x07 | FIFO count register | 0x00 |
| R32_TMR2_COUNT | 0x08 | Current count value register | 0x00000000 |
| R32_TMR2_CNT_END | 0x0C | Count end value register | 0x00000000 |
| R32_TMR2_FIFO | 0x10 | FIFO register | 0x00000000 |
| R32_TMR2_DMA_NOW | 0x14 | Current address of DMA buffer | 0x0000 |
| R32_TMR2_DMA_BEG | 0x18 | Start address of DMA buffer | 0x0000 |
| R32_TMR2_DMA_END | 0x1C | End address of DMA buffer | 0x0000 |

Mode setting register (R8_TMRx_CTRL_MOD) (x=0/1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:6] | RB_TMR_CAP_EDGE | RW | In capture mode, capture trigger mode selection.<br>00: Not triggered;<br>01: Capture the time between any edge changes;<br>10: Capture the time between falling edges;<br>11: Capture the time between rising edges. | 0 |
| [7:6] | RB_TMR_PWM_REPEAT | RW | In PWM mode, data repetition selection:<br>00: Repeat for 1 time; 01: Repeat for 4 times;<br>10: Repeat for 8 times; 11: Repeat for 16 times. | 0 |
| 5 | Reserved | RO | Reserved. | 0 |
| 4 | RB_TMR_CAP_COUNT | RW | Sub-mode of input mode when RB_TMR_MODE_IN=1.<br>1: Counting function; 0: Capture function. | 0 |
| 4 | RB_TMR_OUT_POLAR | RW | In PWM mode, output polarity setting:<br>1: Default high level, active low;<br>0: Default low level, active high; | 0 |
| 3 | RB_TMR_OUT_EN | RW | Timer output enable bit<br>1: Output enabled; 0: Output disabled. | 0 |
| 2 | RB_TMR_COUNT_EN | RW | Timer module enable.<br>1: Enabled; 0: Disabled. | 0 |
| 1 | RB_TMR_ALL_CLEAR | RW | Timer FIFO/counter/interrupt flag clear.<br>1: Forced to empty and clear; 0: No action. | 1 |

| | | | | |
|---|---|---|---|---|
| 0 | RB_TMR_MODE_IN | RW | Timer mode setting.<br>1: Input mode, capture/count function;<br>0: Timing mode/PWM mode | 0 |

Interrupt enable register (R8_TMRx_INTER_EN) (x=0/1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RO | Reserved. | 0 |
| 4 | RB_TMR_IE_FIFO_OV | RW | FIFO overflow (FIFO is full in capture mode or FIFO is empty in PWM mode) interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 3 | RB_TMR_IE_DMA_END | RW | DMA complete interrupt enable (only supported by TMR1/TMR2).<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 2 | RB_TMR_IE_FIFO_HF | RW | FIFO half use (FIFO>=4 in capture mode or FIFO<4 in PWM mode) interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 1 | RB_TMR_IE_DATA_ACT | RW | In capture mode, new data capture interrupt enable.<br>In PWM mode, active level end interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 0 | RB_TMR_IE_CYC_END | RW | In capture mode, capture timeout interrupt enable.<br>In PWM mode, PWM clock cycle end interrupt enable.<br>In timing mode, timing cycle end interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |

Interrupt flag register (R8_TMRx_INT_FLAG) (x=0/1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RO | Reserved. | 0 |
| 4 | RB_TMR_IF_FIFO_OV | RW1 | FIFO overflow (FIFO is full in capture mode or FIFO is empty in PWM mode) flag. Cleared by writing 1.<br>1: Flag triggered; 0: Not triggered. | 0 |
| 3 | RB_TMR_IF_DMA_END | RW1 | DMA complete flag (only supported by TMR1/TMR2).<br>1: Flag triggered; 0: Not triggered. | 0 |
| 2 | RB_TMR_IF_FIFO_HF | RW1 | FIFO half use (FIFO>=4 in capture mode or FIFO<4 in PWM mode) flag. Cleared by writing 1.<br>1: Flag triggered; 0: Not triggered. | 0 |
| 1 | RB_TMR_IF_DATA_ACT | RW1 | In capture mode, new data capture flag. Cleared by writing 1.<br>In PWM mode, active level end flag. Cleared by writing 1.<br>1: Flag triggered; 0: Not triggered. | 0 |
| 0 | RB_TMR_IF_CYC_END | RW1 | In capture mode, capture timeout flag. Cleared by writing 1. | 0 |

| | | | | |
|---|---|---|---|---|
| | | | In PWM mode, PWM clock cycle end flag. Cleared by writing 1.<br>In timing mode, TV cycle end flag. Cleared by writing 1.<br>1: Flag triggered; 0: Not triggered. | |

FIFO count register (R8_TMRx_FIFO_COUNT) (x=0/1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_TMRx_FIFO_COUNT | RO | The data byte count in FIFO, with the maximum value of 8. | X |

Current count value register (R32_TMRx_COUNT) (x=0/1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | R32_TMRx_COUNT | RO | Current count value of counter. | X |

Final count value setting register (R32_TMRx_CNT_END) (x=0/1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | R32_TMRx_CNT_END | RW | In timer mode, the number of clocks in a timing cycle.<br>In PWM mode, the total number of clocks in PWM cycle.<br>In capture mode, the number of captured timeout clock cycles.<br>Only the lower 26 bits are valid, and the maximum value is 67108863.<br>*Note: R32_TMRx_COUNT counts from 0, so the maximum value is R32_TMRx_CNT_END minus 1.* | X |

FIFO register (R32_TMRx_FIFO) (x=0/1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | R32_TMRx_FIFO | RO/ WO | FIFO data register, only the lower 26 bits are valid. | X |

DMA control register (R8_TMRx_CTRL_DMA) (x=1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:3] | Reserved | RO | Reserved. | 0 |
| 2 | RB_TMR_DMA_LOOP | RW | DMA address loop enable (only supported by TMR1/TMR2).<br>1: Address loop; 0: Address once.<br>If the DMA address loop function is enabled, when the DMA address increases to the set end address, | 0 |

| | | | | |
|---|---|---|---|---|
| | | | it automatically loops to the start address set. | |
| 1 | Reserved | RO | Reserved. | 0 |
| 0 | RB_TMR_DMA_ENABLE | RW | DMA function enable (only supported by TMR1/TMR2).<br>1: DMA enabled; 0: DMA disabled. | 0 |

DMA current buffer address (R32_TMRx_DMA_NOW) (x=1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:18] | Reserved | RO | Reserved. | 0 |
| [17:0] | R16_TMRx_DMA_NOW | RW | Current address of DMA data buffer.<br>It can be used to calculate the number of conversions.<br>COUNT=(TMR_DMA_NOW-TMR_DMA_BEG)/4 | X |

DMA start buffer address (R32_TMRx_DMA_BEG) (x=1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:18] | Reserved | RO | Reserved. | 0 |
| [17:0] | R16_TMRx_DMA_BEG | RW | Start address of DMA data buffer, 4 bytes must be aligned for the address.<br>That is, in PWM data transmission or capture mode, the data captured starts from this buffer address. | X |

*Note: This DMA address can cover RAMS and RAMX areas.*

DMA end buffer address (R32_TMRx_DMA_END) (x=1/2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:18] | Reserved | RO | Reserved. | 0 |
| [17:0] | R16_TMRx_DMA_END | RW | End address of DMA data buffer (not included), 4 bytes must be aligned for the address. That is, in PWM data transmission or capture mode, the data captured ends at this buffer address. | X |

*Note: This DMA address can cover RAMS and RAMX areas.*

# Chapter 9 Pulse width modulation (PWMX)

The PWMX module provides 4-channel PWM outputs, with adjustable duty cycle, and the PWM cycle is fixed and 2 modes are available, and the operation is simple. The extended PWM pin outputs are identified as PWM0/PWM1/PWM2/PWM3.

## 9.1 Module configuration

1. Set the PWM0-PWM3 pins direction as output. Optionally, set the drive capability of the corresponding I/Os;
2. Set the R8_PWM_CLOCK_DIV register to calculate the clock reference frequency of PWM;
3. Set the PWM mode control register (R8_PWM_CTRL_MOD), to configure the output polarity of PWMx, and enable the corresponding PWMx (the RB_PWMx_OUT_EN bit set to 1) output;
4. Set the R8_PWM_CTRL_CFG register and R32_PWM_DATA register to configure the PWM duty cycle output.

Calculation: PWMx duty cycle = R8_PWMx_DATA / (RB_PWM_CYCLE_SEL? 255: 256)

*Note: If the corresponding RB_PWMx_OUT_EN output enable is always on in the R8_PWM_CTRL_MOD register, the PWM waveform will be output continuously until RB_PWMx_OUT_EN is disabled.*

## 9.2 Register description

PWMX register physical base address: 0x40005000

Table 9-1 PWM0/1/2/3 registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_PWM_CTRL_MOD | 0x00 | PWM mode control register | 0x00 |
| R8_PWM_CTRL_CFG | 0x01 | PWM configuration control register | 0x00 |
| R8_PWM_CLOCK_DIV | 0x02 | PWM clock divider register | 0x00 |
| R32_PWM_DATA | 0x04 | PWM0/1/2/3 data hold register | X |
| R8_PWM0_DATA | 0x04 | PWM0 data hold register | X |
| R8_PWM1_DATA | 0x05 | PWM1 data hold register | X |
| R8_PWM2_DATA | 0x06 | PWM2 data hold register | X |
| R8_PWM3_DATA | 0x07 | PWM3 data hold register | X |

PWM mode control register (R8_PWM_CTRL_MOD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_PWM3_POLAR | RW | PWM3 output polarity control.<br>1: Default high level, active low;<br>0: Default low level, active high; | 0 |
| 6 | RB_PWM2_POLAR | RW | PWM2 output polarity control.<br>1: Default high level, active low;<br>0: Default low level, active high; | 0 |
| 5 | RB_PWM1_POLAR | RW | PWM1 output polarity control.<br>1: Default high level, active low;<br>0: Default low level, active high; | 0 |

| | | | | |
|---|---|---|---|---|
| 4 | RB_PWM0_POLAR | RW | PWM0 output polarity control.<br>1: Default high level, active low;<br>0: Default low level, active high; | 0 |
| 3 | RB_PWM3_OUT_EN | RW | PWM3 output enable.<br>1: Enabled; 0: Disabled. | 0 |
| 2 | RB_PWM2_OUT_EN | RW | PWM2 output enable.<br>1: Enabled; 0: Disabled. | 0 |
| 1 | RB_PWM1_OUT_EN | RW | PWM1 output enable.<br>1: Enabled; 0: Disabled. | 0 |
| 0 | RB_PWM0_OUT_EN | RW | PWM0 output enable.<br>1: Enabled; 0: Disabled. | 0 |

PWM configuration control register (R8_PWM_CTRL_CFG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:1] | Reserved | RW | Reserved. | 0 |
| 0 | RB_PWM_CYCLE_SEL | RW | PWM cycle selection.<br>1: 255 clocks; 0: 256 clocks. | 0 |

PWM clock divider register (R8_PWM_CLOCK_DIV)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | R8_PWM_CLOCK_DIV | RW | PWM reference clock frequency division factor.<br>Calculation: CLK=Fsys/R8_PWM_CLOCK_DIV. | 0 |

PWM0/1/2/3 data hold register (R32_PWM_DATA)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:24] | R8_PWM3_DATA | RW | PWM3 data hold register. | X |
| [23:16] | R8_PWM2_DATA | RW | PWM2 data hold register. | X |
| [15:8] | R8_PWM1_DATA | RW | PWM1 data hold register. | X |
| [7:0] | R8_PWM0_DATA | RW | PWM0 data hold register. | X |

# Chapter 10 High speed parallel interface (HSPI)

Parallel interface refers to multiple data lines used to transfer data in parallel. The system is equipped with a set of high-speed parallel interface (HSPI) to implement high-speed transfer in half-duplex mode of the interface through 8 receive/transmit control signal lines and 32 data lines. The speed can reach 3.8Gbps, which can be used for the fast transfer of large amount of data, customized high-speed transfer and other applications.

## 10.1 Main features

- Programmable 8-bit, 16-bit and 32-bit data width
- Provide hardware automatic response mode
- Provide burst transfer mode
- Based on the data packet receive transmit structure, which is convenient for application expansion
- Built-in multi-level 128-bit FIFO, DMA transfer
- Reliable transfer: including CRC check, receive/transmit serial number
- Support double buffering transceiver mechanism
- Up to 3.8Gbps transfer speed

## 10.2 Functional specification

HSPI defines the transmission interface for 8/16/32-bit parallel data high-speed communication and a simple and reliable interactive protocol. The module implements fast transfer of interface data through built-in multi-level transceiver FIFO, burst mode, DMA mechanism, etc. At the same time, the reliability of transfer is guaranteed by the data packet structure, response mechanism and transceiver sequence, which is simple and efficient.

### 10.2.1 Hardware interface description

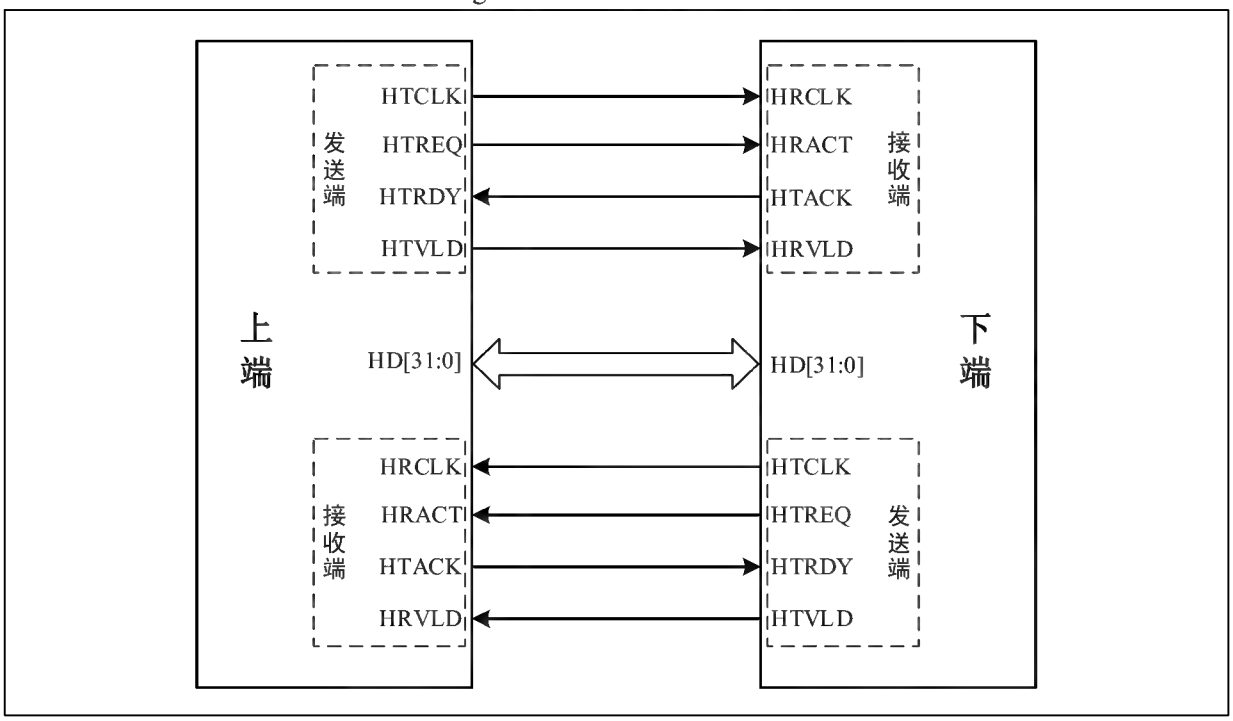
HSPI hardware includes: 8 control signal lines and 32 data signal lines, as described in Table 10-1. Among them, the control signals are divided into transmission end and reception end, with 4 control lines for each. When the module is operating, transmission end control signal of one line should be connected to the reception end control signal of the other line, as shown in Figure 10-1. Since there is only 1 set of data lines, HSPI can only be in half-duplex communication mode. When the lines request transmission at the same time, the module will arbitrate the priority of data transmission according to whether the current configuration is the upper mode or the lower mode.

Table 10-1 HSPI hardware interface

| HSPI definition | | | IO mode configuration | Description |
|---|---|---|---|---|
| Control line | Tx | HTCLK | Push-pull output | Communication clock signal is output, and the clock frequency comes from the system clock frequency. |
| | | HTREQ | Push-pull output | Output transmit request signal, active high. Connected to the HRACT pin. |
| | | HTRDY | Pull-down input | Detect the status of reception end. |
| | | HTVLD | Push-pull output | Output data transmit status, active high. Connected to the HRVLD pin. |

| | | | | |
|---|---|---|---|---|
| | Rx | HRCLK | Pull-down input | Input the sampling clock as the sampling clock reference for receiving data. |
| | | HRACT | Pull-down input | Input transmit request signal, and then drive the HTACK signal after it is active. |
| | | HTACK | Push-pull output | Output ready receive status signal, active high. Connected to the HTRDY pin. |
| | | HRVLD | Pull-down input | Detect HTVLD pin status. |
| Data line | Bidirectional | HD0～HD31 | Floating input | Parallel data port, supports 8-bit, 16-bit and 32-bit data formats. |

Figure 10-1 HSPI connection




### 10.2.2 Data packet transfer format

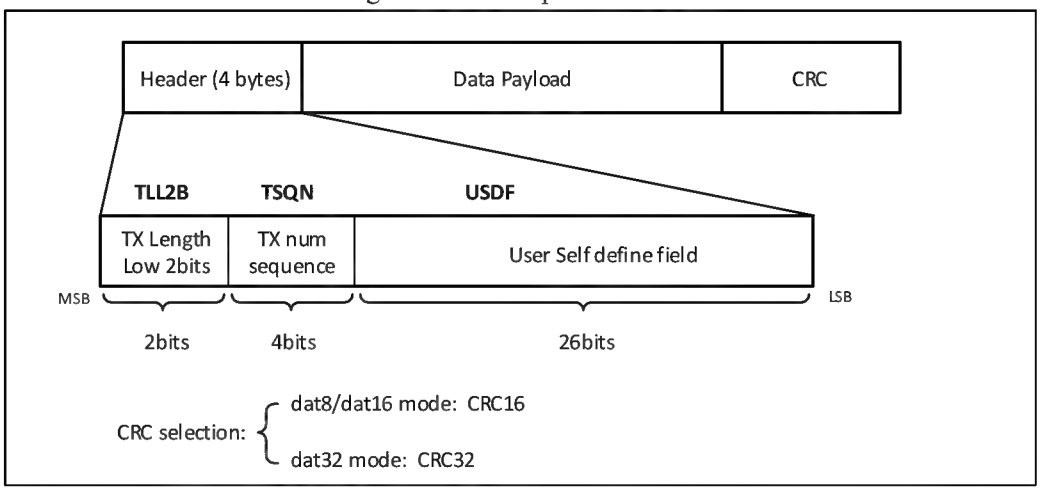
The HSPI transfer takes data packet as the unit, and the low 8/16/32 bits are transferred first. A complete data packet structure includes:

1) 4-byte packet header, including TX length low 2bits (TLL2B), 4-bit TX sequence number (TSQN), 26-bit user self-defined field (USDF)

2) Data payload with configurable length

3) CRC code, 2-byte CRC16 or 4-byte CRC32

Figure 10-2 Data packet format




- Header: 32-bit, composed of TLL2B, TSQN and USDF, which is fixedly added by hardware during transmission. The TLL2B indicates the valid bit width occupied by the payload of data in the last data line width in current data packet being transmitted (transmission length and the data line bit width are not aligned), which is automatically processed by hardware, and application code is not required. The TSQN indicates the sequence number (0-15) of the current data packet being transmitted. After each successful transmission, hardware automatically performs the increment of the next serial number. This can ensure the continuity of data packet transmission and reception, corresponding to the RB_HSPI_TX_NUM and RB_HSPI_RX_NUM bits in the register. The USDF is reserved for user to self-define the extended functions and can be used to fill the command word, address field and time stamp information of the upper-layer protocol, corresponding to the HSPI_UDF0/HSPI_UDF1 register.

- Data payload: Mass data transfer area. At the transmission end, the application program can configure its transmission length with range (1-4096) and the SRAM address of the data to be transmitted (HSPI_TX_ADDR0/HSPI_TX_ADDR1). At the reception end, the application code can configure the SRAM address of the data to be received (HSPI_RX_ADDR0/HSPI_RX_ADDR1) and receive the payload data length in the reception length register.

*Note: HSPI module DMA addresses RAMX area, the lowest address is 0x20020000, pay attention to address allocation for application program.*

- CRC code: Check result of packet header + data payload, to ensure the reliability of transmission. When the transmission data width is configured as 8/16-bit data mode, use CRC16 polynomial (0x8005): $X^{16}+X^{15}+X^{2}+1$. When the transmission data width is configured as 32-bit data mode, use CRC32 polynomial (0x4C11DB7): $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$.

### 10.2.3 Transfer configuration and procedure

When describing the transfer procedure of the HSPI interface below, the interface at one end with priority in transmission is called the "upper end", and the interface at the other end is called the "lower end". Both ends can implement transmission and reception, and the transmission and reception methods at both ends are the same. The interface transmission is triggered by software application code. For reception, hardware automatically samples and receives after the module starts running.

- HSPI interface initialization

Controlled by the reset signal inside the HSPI interface (in the HSPI_CTRL register):

1) RB_HSPI_ENABLE: Module reference clock output (HTCLK) and internal logic operation enable bit.

2) RB_HSPI_TRX_RST: Reset the digital logic circuit and state of HSPI module, to resume the module to its initial working state.
3) RB_HSPI_ALL_CLR: HSPI internal FIFO data and interrupt reset flag.
*Note: These 3 control bits do not change the configuration value of HSPI register.*

Initialization procedure:
1) Set the IO port mode of HSPI interface pin as described in Section 10.2.1.
2) Set the upper/lower end modes and data width of the current interface of HSPI_CFG register.
3) Optionally, set the HSPI_CFG register, to configure the hardware response function and dual transmit/receive buffer functions.
4) Set the HSPI_AUX register to transmit/receive sampling edge.
5) Optionally, set the HSPI_INT_EN register, so that the interface can trigger the HSPI interrupt under the corresponding conditions.
6) Configure the HSPI_CTRL register, clear the RB_HSPI_ALL_CLR and RB_HSPI_TRX_RST bits, and set RB_HSPI_ENABLE and RB_HSPI_DMA_EN bits to 1, to enable HSPI function and internal DMA.

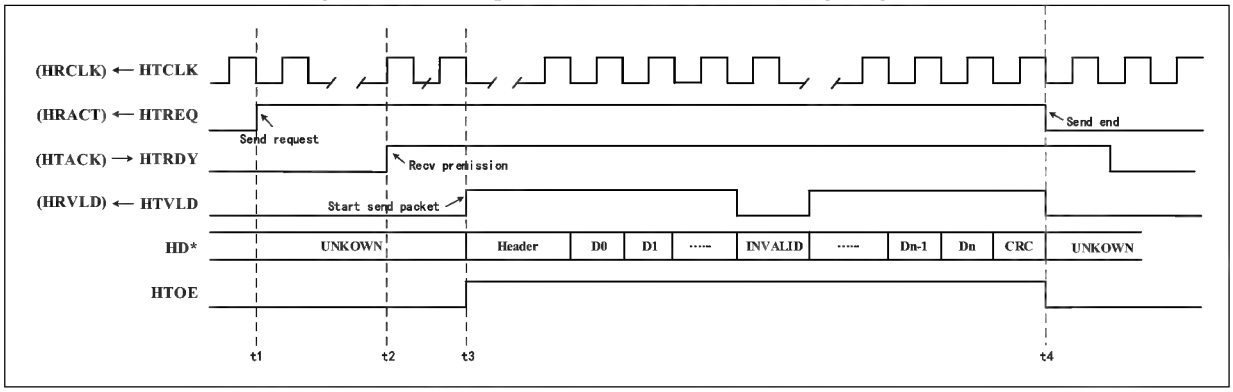
- Single packet data reception/transmission

Take the transmission at upper end and the reception at lower end as an example. When transmitting at upper end, first the hardware drives the transmission control line HTREQ pin to output high level, to inform the lower end that there is data to be transmitted. In this case, if transmission is allowed at the lower end, its hardware will drive the reception control line HTACK (HTRDY) to high level output, otherwise, it will keep at low level. After the upper end reads the valid level on HTRDY, it starts to prepare the data to be transmitted. The internal DMA starts to shift the data from the SRAM to the built-in FIFO of the module, which lasts for at least 10 clock cycles, the specific time depends on the current system bus bandwidth allocation. After that, the HTVLD pin signal is set and the parallel data line control right is taken over, and the data packet is transmitted according to the parallel bit width and clock transmit edge configured by the module. The data packet format is described in Section 10.2.2.

If the DMA fails to fill data into the FIFO in time due to the internal scheduling of the module during the transmission, the FIFO is empty, and the HTVLD signal is pulled down. In this case, the data on the bus is invalid. After there is data in the FIFO, HTVLD will be set again and continue to transmit data. When all the data bytes are transmitted, the upper hardware drives the HTVLD/HTREQ signal and pull it down. At this time, a complete data packet is transmitted. Figure 10-3 is the receive/transmit timing diagram. To transmit a new packet of data, repeat the above process.

When receiving at the lower end, HRCLK inputs clock signal as the data sampling clock reference, and the software can set the data sampling edge. When the HRVLD signal is valid (high level), the receiver will sample the parallel data at the valid edge of clock and save it in the internal FIFO, and the data will be transferred from the DMA to the receiving SRAM area configured by the software, and alternate storage methods of dual DMA buffers will be supported. When the HRACT signal is invalid, it is considered that a data packet has been received at the lower end, it is required to check the CRC value and serial number, record the effective data length and user-defined fields to the corresponding register, and report the receiving flag status to the application code.

Figure 10-3 Data packet receive/transmit timing diagram




Before starting transmission, it is required to configure the data packet to be transmitted:

1) DMA transmit address: Set the HSPI_TX_ADDR0 register. If in dual buffering mode, it is also required to set the HSPI_TX_ADDR1 register.
2) Transmission packet sequence number: Set the HSPI_TX_SC register. If in dual buffering mode, DMA start access address can be set.
3) User self-define field: Set either the HSPI_UDF0 register or the HSPI_UDF1 register, which depends on the parity of transmit sequence number.
4) Transmission length: Set HSPI_DMA_LEN0 register. To adopt double buffering mode, it is also required to set HSPI_DMA_LEN1 register.

The software starts transmission by setting the RB_HSPI_SW_ACT bit to 1. After that, it is needed to check the transmission result by query or interruption, and judge whether the transmission of single packet has been completed (HTREQ invalid) by accessing the RB_HSPI_IF_T_DONE bit in the HSPI_INT_FLAG register.

Before preparing to receive, it is required to configure the storage of the data packet to be received:

1) DMA receive address: Set the HSPI_RX_ADDR0 register. If in dual buffering mode, it is required to set the HSPI_RX_ADDR1 register.
2) Reception sequence number: Stored in the HSPI_RX_SC register. If in dual buffering mode, the initial receive address of DMA can be set;
3) User self-define field: Set either the HSPI_UDF0 register or the HSPI_UDF1 register, which depends on the parity of receive sequence number.

After the hardware sets the RB_HSPI_IF_R_DONE flag, the data packet is received. In this case, read the RB_HSPI_NUM_MIS and RB_HSPI_CRC_ERR bits in the HSPI_RTX_STATUS register, to determine whether CRC check and sequence number are correct. If correct, the current data packet is valid, and the valid length of data packet is obtained through HSPI_RX_LEN0 register. If in dual buffering mode, the HSPI_RX_LEN1 register is also required.

- Hardware auto acknowledge

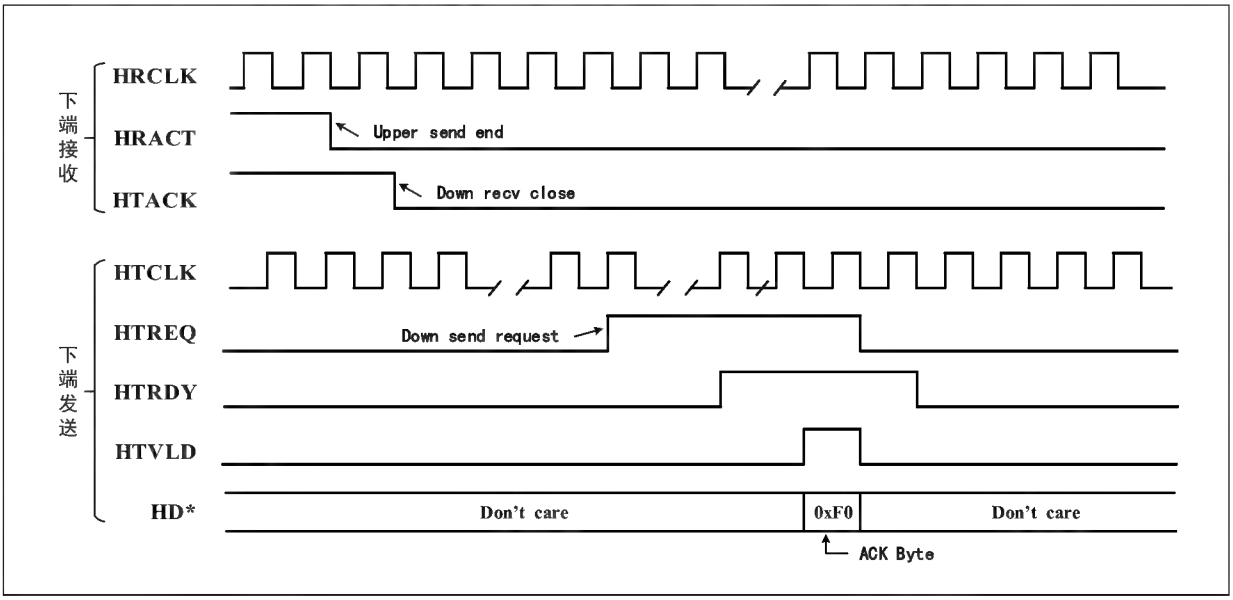
Hardware auto acknowledge is a response from the reception end to the correctness and continuity of the data packet transmitted by the transmission end. It is enabled by setting the HSPI_HW_ACK bit in the HSPI_CFG register to 1.

In this mode, when the reception end receives a complete data packet (HTACK invalid), if the CRC check is correct and the reception sequence number is matched, the hardware will automatically transmit a special response byte (0xF0) to indicate that the current data packet has been successfully received. Otherwise, no

response will be sent, and response timeout state will be queried at the transmission end.

The above process of transmission response does not require the application program. The transmission end automatically monitors the response identifier after the transmission is completed, and the reception end automatically transmits the response byte after a packet is received.

Figure 10-4 Hardware auto-acknowledge timing diagram




- DMA dual buffering

The module provides 2 sets of registers for the reception/transmission address and length, which are mainly used in dual buffering mode of DMA. By setting the RB_HSPI_RX_TOG_EN and RB_HSPI_TX_TOG_EN bits in the HSPI_CFG register, the DMA dual buffering function for reception and transmission can be enabled respectively. In this case, every time a data packet is successfully transmitted or received, the address and length register access of module will be shifted to another set of registers. The RB_HSPI_TX_TOG and RB_HSPI_RX_TOG bits indicate the currently used reception and transmission register flags respectively.

Through DMA dual buffering mode, the application code can process data of another set of buffers when the hardware is performing reception and transmission operations on a set of buffers, so that the hardware reception and transmission and software processing can be synchronized to improve data transmission efficiency.

*Note: If the dual buffering mode is not used, but the burst transmission mode is enabled, the DMA address and length must be updated in time, otherwise, data packets will be transmitted repeatedly. In the timing sequence diagram of Figure 10-3, the address and length cannot be modified within t1-t3. At this time, the current register is still used inside the module, and it can be modified and switched within t3-t4.*

- Transmission priority

HSPI interface can receive and transmit data. While it can only transmit or receive at some point (half-duplex transfer mode), as data line is shared. According to the "Single-Packet Data receive/transmit timing ", the transmit request is triggered by the valid level on HTREQ, so when both ends of the HSPI interface drive HTREQ pin, the end at high level has priority to obtain the transmit authority. If both ends drive the HTREQ pin at the same time, the end configured in the "upper" mode has the priority to obtain the transmit authority. Configure the HSPI interface to be in "upper" or "lower" mode through the RB_HSPI_MODE bit in the HSPI_CFG register.

- Burst mode transmission

Since software needs to participate in providing a trigger in the process of transmitting each data packet, HSPI interface will provide "burst mode transmission" to improve transmission efficiency, and the hardware will automatically send a specified number of data packets without human intervention. The total number of data packets (RB_HSPI_BURST_LEN) that need to be configured by application code, and 256 continuous data packets are supported at most. This mode is enabled by setting the RB_HSPI_BURST_EN bit in the HSPI_BURST_CFG register to 1. In the burst mode, after the software trigger signal is obtained (RB_HSPI_SW_ACT bit is 1), the hardware will send the first data packet according to the address and length register, to meet the "single packet data receive/transmit timing". After a single packet has been transmitted, the transmission of the next data packet will be enabled immediately. There is no need to wait for the software trigger signal until the specified number of data packets have been transmitted. The application code can read the HSPI_BURST_CNT register to get the number of current data packets that have been successfully sent, and the hardware will automatically reset it after the transmission is completed. In burst mode, the hardware will set RB_HSPI_IF_B_DONE flag when the transmission is completed or in the process of waiting for a response timeout, and the burst mode function will be disabled (RB_HSPI_BURST_EN=0). To continue using burst mode, software is needed to restart. In burst mode transmission, the hardware will set the RB_HSPI_IF_T_DONE interrupt flag after each independent data packet transmission is completed, but the transmission/reception sequence number will increase only after the transmission is completed successfully.

If the hardware auto-acknowledge function is enabled, the transmission end will start the next data packet transmission after each single packet has been transmitted and the response byte has been received. The current burst transmission will be suspended in case of response timeout or response error.

If DMA dual buffering function is enabled, the transmission/reception address and length register will be converted after a data packet is successfully transmitted or received.

*Note 1: When multiple consecutive data packets are transmitted, the parity status of TSQN determines the register corresponding to USDF, regardless of whether dual buffering is enabled.*

*Note 2: If the "Transmission Priority" selection occurs during the data packet transmission interval in burst mode, the burst mode transmission will be suspended, and priority transmission processing will be performed before continuing transmission.*

*Note 3: It is required to enable the hardware auto-acknowledge mode before using the burst mode, to confirm that multiple consecutive packets have been successfully transmitted.*

## 10.4 Register description

HSPI peripheral register base address: 0x40006000

Table 10-1 HSPI registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_HSPI_CFG | 0x00 | HSPI configuration register | 0x82 |
| R8_HSPI_CTRL | 0x01 | HSPI control register | 0x18 |
| R8_HSPI_INT_EN | 0x02 | HSPI interrupt enable register | 0x00 |
| R8_HSPI_AUX | 0x03 | HSPI auxiliary register | 0x00 |
| R32_HSPI_TX_ADDR0 | 0x04 | HSPI transmission address register 0 | 0x0000000 |
| R32_HSPI_TX_ADDR1 | 0x08 | HSPI transmission address register 1 | 0x0000000 |
| R32_HSPI_RX_ADDR0 | 0x0C | HSPI reception address register 0 | 0x0000000 |

| | | | |
|---|---|---|---|
| R32_HSPI_RX_ADDR1 | 0x10 | HSPI reception address register 1 | 0x0000000 |
| R16_HSPI_DMA_LEN0 | 0x14 | HSPI transmission length register 0 | 0x0000 |
| R16_HSPI_RX_LEN0 | 0x16 | HSPI reception length register 0 | 0x0000 |
| R16_HSPI_DMA_LEN1 | 0x18 | HSPI transmission length register 1 | 0x0000 |
| R16_HSPI_RX_LEN1 | 0x1A | HSPI reception length register 1 | 0x0000 |
| R16_HSPI_BURST_CFG | 0x1C | HSPI burst mode configuration register | 0x0000 |
| R8_HSPI_BURST_CNT | 0x1E | HSPI burst mode counter | 0x00 |
| R32_HSPI_UDF0 | 0x20 | HSPI user-defined field 0 | 0x0000000 |
| R32_HSPI_UDF1 | 0x24 | HSPI user-defined field 1 | 0x0000000 |
| R8_HSPI_INT_FLAG | 0x28 | HSPI interrupt flag register | 0x00 |
| R8_HSPI_RTX_STATUS | 0x29 | HSPI receive/transmit status register | 0x00 |
| R8_HSPI_TX_SC | 0x2A | HSPI transmission sequence control register | 0x00 |
| R8_HSPI_RX_SC | 0x2B | HSPI reception sequence control register | 0x00 |

HSPI configuration register (R8_HSPI_CFG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_HSPI_HW_ACK | RW | Acknowledge mode configuration.<br>1: Hardware auto-acknowledge mode;<br>0: No acknowledge. | 1 |
| 6 | RB_HSPI_RX_TOG_EN | RW | Dual buffering receive enable.<br>1: The buffer is received alternately by address 0 and address 1;<br>0: The buffer is received by address 0. | 0 |
| 5 | RB_HSPI_TX_TOG_EN | RW | Dual buffering transmit enable.<br>1: The buffer is transmitted alternately by address 0 and address 1;<br>0: The buffer is transmitted by address 0. | 0 |
| 4 | Reserved | RO | Reserved. | 0 |
| [3:2] | RB_HSPI_MSK_SIZE | RW | Parallel data size.<br>00: 8-bit mode;<br>01: 16-bit mode;<br>1x: 32-bit mode. | 0 |
| 1 | RB_HSPI_DUDMA | RW | Internal dual DMA mode enable. The system DMA bandwidth of HSPI interface can be doubled by enabling this function.<br>1: Dual DMA request enabled;<br>0: Dual DMA request disabled. | 1 |
| 0 | RB_HSPI_MODE | RW | Transmission priority configuration. When both ends of HSPI interface transmit requests at the same time, the upper end has priority.<br>1: Upper end mode (priority);<br>0: Lower end mode. | 0 |

HSPI control register (R8_HSPI_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RO | Reserved. | 0 |
| 4 | RB_HSPI_TRX_RST | RW | HSPI module receive/transmit logic reset control.<br>1: Internal logic unit reset;<br>0: No action. | 1 |
| 3 | RB_HSPI_ALL_CLR | RW | HSPI module built-in FIFO/interrupt flag reset control.<br>1: FIFO and interrupt flag reset;<br>0: No action. | 1 |
| 2 | RB_HSPI_SW_ACT | RW | Software triggers the transmission of data packets. Set to 1 by software, and cleared by hardware.<br>1: A data packet transmission triggered;<br>0: No action. | 0 |
| 1 | RB_HSPI_DMA_EN | RW | DMA enable.<br>1: DMA enabled; 0: DMA disabled. | 0 |
| 0 | RB_HSPI_ENABLE | RW | HSPI enable.<br>1: HSPI enabled; 0: HSPI disabled. | 0 |

HSPI interrupt enable register (R8_HSPI_INT_EN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:4] | Reserved | RO | Reserved. | 0 |
| 3 | RB_HSPI_IE_B_DONE | RW | In burst mode, the burst sequence transmission complete interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 2 | RB_HSPI_IE_FIFO_OV | RW | Receive/transmit FIFO overflow interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 1 | RB_HSPI_IE_R_DONE | RW | Single packet reception complete interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |
| 0 | RB_HSPI_IE_T_DONE | RW | Single packet transmission complete interrupt enable.<br>1: Interrupt enabled; 0: Interrupt disabled. | 0 |

HSPI auxiliary register (R8_HSPI_AUX)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RO | Reserved. | 0 |
| [4:3] | RB_HSPI_ACK_CNT_SEL | RW | In delayed transmission ACK mode, delay time control:<br>00: Delay 2 $T_{sys}$ to transmit ACK;<br>01: Delay 4 $T_{sys}$ to transmit ACK;<br>10: Delay 6 $T_{sys}$ to transmit ACK;<br>11: Delay 8 $T_{sys}$ to transmit ACK. | 0 |

| | | | | |
|---|---|---|---|---|
| 2 | RB_HSPI_ACK_TX_MOD | RW | Hardware auto ACK transmission time configuration:<br>1: When the HRVLD signal becomes invalid, delay transmitting ACK;<br>0: When all data is written into SRAM, immediately transmit ACK. | 0 |
| 1 | RB_HSPI_RCK_MOD | RW | Receive data sampling edge configuration.<br>1: Sample data on the falling edge of clock;<br>0: Sample data on the rising edge of clock. | 0 |
| 0 | RB_HSPI_TCK_MOD | RW | Transmit data clock polarity configuration.<br>1: Transmit data on the falling edge of clock;<br>0: Transmit data on the rising edge of clock; | 0 |

HSPI transmission address 0 register (R32_HSPI_TX_ADDR0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | RB_HSPI_TX_ADDR0 | RW | Configure DMA transmission address 0. The lower 4 bits are fixed to 0 (16 bytes are aligned). | 0 |

*Note: DMA addresses RAMX area.*

HSPI transmission address 1 register (R32_HSPI_TX_ADDR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | RB_HSPI_TX_ADDR1 | RW | Configure DMA transmission address 1. The lower 4 bits are fixed to 0 (16 bytes are aligned).<br>*Note: Used in dual buffering mode.* | 0 |

*Note: DMA addresses RAMX area.*

HSPI reception address 0 register (R32_HSPI_RX_ADDR0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | RB_HSPI_RX_ADDR0 | RW | Configure DMA reception address 0. The lower 4 bits are fixed to 0 (16 bytes are aligned). | 0 |

*Note: DMA addresses RAMX area.*

HSPI reception address 1 register (R32_HSPI_RX_ADDR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | RB_HSPI_RX_ADDR1 | RW | Configure DMA reception address 1. The lower 4 bits are fixed to 0 (16 bytes are aligned).<br>*Note: Used in dual buffering mode.* | 0 |

*Note: DMA addresses RAMX area.*

HSPI DMA transmission length 0 register (R16_HSPI_DMA_LEN0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:12] | Reserved | RO | Reserved. | 0 |
| [11:0] | RB_HSPI_DMA_LEN0 | RW | Configure DMA transmission length 0. Actual transmission length (LEN0+1). | 0 |

HSPI reception length 0 register (R16_HSPI_RX_LEN0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:12] | Reserved | RO | Reserved. | 0 |
| [11:0] | RB_HSPI_RX_LEN0 | RW | Configure DMA reception length 0. Data 0 means 4096 bytes. | 0 |

HSPI DMA transmission length 1 register (R16_HSPI_DMA_LEN1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:12] | Reserved | RO | Reserved. | 0 |
| [11:0] | RB_HSPI_DMA_LEN1 | RW | Configure DMA transmission length 1. Actual transmission length (LEN0+1).<br>*Note: Used in dual buffering mode.* | 0 |

HSPI reception length 1 register (R16_HSPI_RX_LEN1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:12] | Reserved | RO | Reserved. | 0 |
| [11:0] | RB_HSPI_RX_LEN1 | RW | Configure DMA reception length 1. Value 0 means 4096 bytes.<br>*Note: Used in dual buffering mode.* | 0 |

HSPI burst configuration register (R16_HSPI_BURST_CFG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:8] | RB_HSPI_BURST_LEN | RW | Configure the number of burst transmission packets. | 0 |
| [7:1] | Reserved | RO | Reserved. | 0 |
| 0 | RB_HSPI_BURST_EN | RW | Burst transfer enable. Set to 1 by software, automatically cleared by hardware when an error occurs after the burst transfer is completed.<br>1: Burst transfer enabled;<br>0: Burst mode disabled. | 0 |

HSPI burst counter register (R8_HSPI_BURST_CNT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:0] | RB_HSPI_BURST_CNT | RO | Burst transfer counter. Record the number of packets that have been successfully transmitted in the current burst mode. The operation of enabling burst automatically resets the counter, and the hardware automatically resets the counter after the number of burst setting packets is reached. | 0 |

HSPI user-defined field 0 register (R32_HSPI_UDF0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:26] | Reserved | RO | Reserved. | 0 |
| [25:0] | RB_HSPI_UDF0 | RW | When transmitting, the software sets the customized contents to be transmitted;<br>When receiving, store the customized contents received in the data packet.<br>*Note: Select this register when the serial number in the reception/transmission sequence control register is an even number.* | 0 |

HSPI user-defined field 1 register (R32_HSPI_UDF1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:26] | Reserved | RO | Reserved. | 0 |
| [25:0] | RB_HSPI_UDF1 | RW | When transmitting, software sets the customized contents to be transmitted;<br>When receiving, store the customized contents received in the data packet.<br>*Note: Select this register when the serial number in the reception/transmission sequence control register is an odd number.* | 0 |

HSPI interrupt flag register (R8_HSPI_INT_FLAG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:4] | Reserved | RO | Reserved. | 0 |
| 3 | RB_HSPI_IF_B_DONE | RW1 | In burst mode, burst sequence packet transmission complete flag.<br>1: Burst transmission completed;<br>0: No event. | 0 |
| 2 | RB_HSPI_IF_FIFO_OV | RW1 | Receive/transmit FIFO overflow flag.<br>1: Receive/transmit FIFO overflow;<br>0: No event. | 0 |

| | | | | |
|---|---|---|---|---|
| 1 | RB_HSPI_IF_R_DONE | RW1 | Single packet reception complete flag.<br>1: Data reception completed; 0: No event.<br>*Note: After the completion of reception, it is required to read the status register to determine whether the received data CRC is correct.* | 0 |
| 0 | RB_HSPI_IF_T_DONE | RW1 | Single packet transmission complete flag.<br>1: Data transmission completed; 0: No event.<br>*Note: In the hardware auto-acknowledge mode, it is required to read the status register to determine whether the transmission status is successful.* | 0 |

*Note: Even if the R8_HSPI_INT_EN register is not configured, the flags are still set when the conditions are met.*

HSPI receive/transmit status register (R8_HSPI_RTX_STATUS)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:3] | Reserved | RO | Reserved. | 0 |
| 2 | RB_HSPI_NUM_MIS | RO | Reception sequence number status.<br>1: Not matched; 0: Matched. | 0 |
| 1 | RB_HSPI_CRC_ERR | RO | Reception CRC check status.<br>1: CRC check error;<br>0: CRC check correct. | 0 |
| 0 | Reserved | RO | Reserved. | 0 |

HSPI transmission sequence control register (R8_HSPI_TX_SC)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RO | Reserved. | 0 |
| 4 | RB_HSPI_TX_TOG | RWT | Transmission flag.<br>1: DMA transmission address 1, length register 1;<br>0: DMA transmission address 0, length register 0.<br>In dual buffering mode, hardware automatically toggles this bit when the transmission is completed successfully. Software writes 1 to perform toggling, and it is invalid when writing 0.<br>*Note: Act in dual buffering mode.* | 0 |
| [3:0] | RB_HSPI_TX_NUM | RW | Transmission sequence number.<br>After the transmission is completed successfully, the sequence number is automatically incremented and the software can set it.<br>*Note: This domain determines the TSQN field of transmission packet.* | 0 |

*Note: In hardware auto-acknowledge mode, the transmission flag is toggled and the transmission sequence number is incremented only when the acknowledge packet is received correctly. In the non-hardware auto-acknowledge mode, the transmission flag is toggled and the transmission sequence number is incremented when the transmission is completed (acknowledge not judged).*

HSPI reception sequence control register (R8_HSPI_RX_SC)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RO | Reserved. | 0 |
| 4 | RB_HSPI_RX_TOG | RWT | DMA reception address flag.<br>1: DMA address 1;<br>0: DMA address 0.<br>When the reception is completed, if the CRC check is correct and the transmission sequence number matches, hardware automatically toggles this bit.<br>Software writes 1 to perform toggling, and it is invalid when writing 0.<br>*Note: Act in dual buffering mode.* | 0 |
| [3:0] | RB_HSPI_RX_NUM | RW | Reception sequence number.<br>When reception is completed, the reception sequence number is incremented and can be set by software if the CRC check is correct and the reception/transmission sequence number matches.<br>*Note: This domain is used to compare the TSQN field of the received packets.* | 0 |

# Chapter 11 External bus interface (BUS8)

The system is equipped with an external bus controller interface, which supports 32KB of addressing space, and the address ranges from 0x80000000 to 80007FFF. The interface includes single read signal pin (PRD#, namely PA8), single write signal pin (PWR#, namely PA9), 15-bit address pins (PB0-PB14) and 8-bit data pins (PA0-PA7). When the external bus interface is not enabled, the above pins can be used as general I/O ports. The interface supports static memory mapping components, including RAM, ROM, Flash, and some external I/O components, and it can automatically modify the pulse width of read/write signals as well as the setup time and hold time of address and data.

## 11.1 Main features

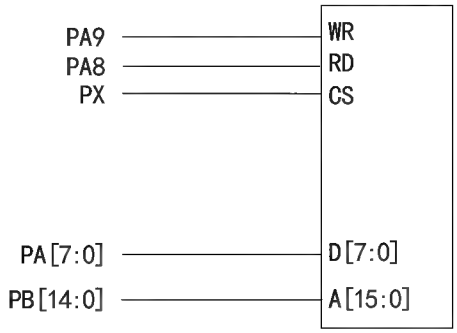
- Support 8-bit bus width
- Supports up to 15-bit address bus width
- Support static memory mapping components, including RAM, Flash and some external I/O components
- Support dynamically modify read/write signal pulse width, setup time and hold time of address and data, etc.

## 11.2 External bus interface application

### 11.2.1 External bus interface

The external bus interface includes single read signal pin PRD# (PA8), single write signal pin PWR# (PA9), 15 address pins A0-A14 (PB0-PB14) and 8 data pins D0-D7 (PA0-PA7). Figure 11-1 shows the way for this interface to connect to external components.

Figure 11-1 BUS8 interface connected to an external chip




### 11.2.2 Bus timing

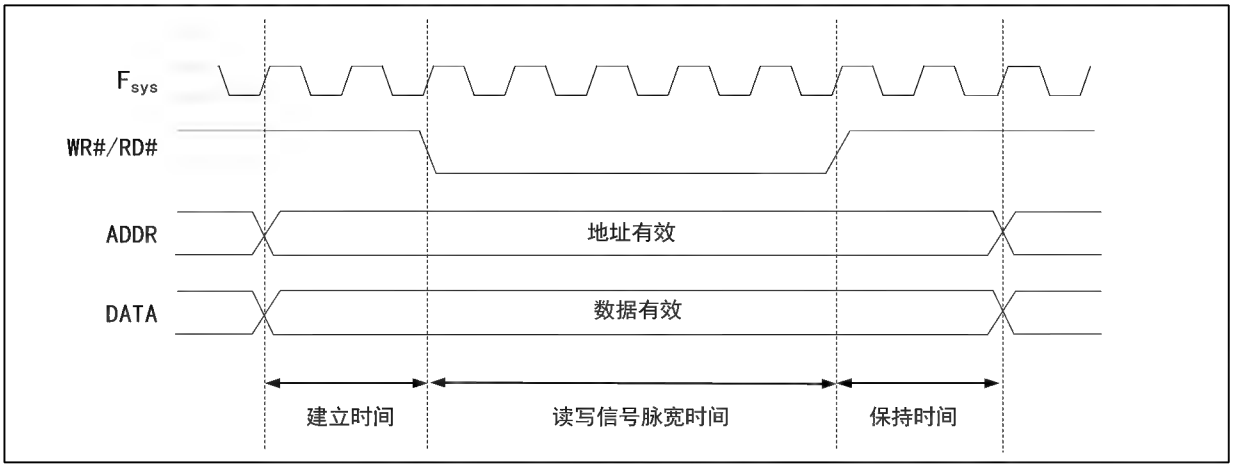
When the BUS8 bus is working, some timing parameters in communication timing are adjustable, such as data setup time, read/write valid time and data hold time, by configuring the [7:4] bits in the R8_XBUS_CONFIG register. Data setup time can be configured by the RB_XBUS_SETUP bit. When it is set to 0, the external bus setup time is 2 clock cycles (Tsys). When it is set to 1, the external bus setup time is 3 clock cycles (Tsys). Data hold time can be configured by the RB_XBUS_HOLD bit. When it is set to 0, the data hold time is 2 clock cycles (Tsys). When it is set to 1, the data hold time is 3 clock cycles (Tsys). The effective window width of bus data read/write can be configured by RB_XBUS_WIDTH, and 3 clocks, 5 clocks, 9 clocks or 16 clocks are for selection.

The address bus width is optional (6-bit/10-bit/15-bit). The unused address lines can be used for other

functions.

Figure 11-2 shows the communication timing when RB_XBUS_WIDTH=01b, RB_XBUS_HOLD=0b and RB_XBUS_SETUP=0b. $F_{sys}$ is the bus clock frequency. RD# is the read control signal, and WR# is the write control signal.

Figure 11-2 BUS8 bus interface read/write access timing diagram




### 11.2.3 External bus configuration

1. Configure the external bus configuration register (R8_XBUS_CONFIG), set the RB_XBUS_ENABLE bit to 1 to enable the external bus interface, and set the address bit control domain (RB_XBUS_ADDR_OE) as needed to select the required addressing range;
2. Configure the external bus configuration register (R8_XBUS_CONFIG), set the RB_XBUS_WIDTH domain to select the bus read/write clock pulse width. Set the RB_XBUS_SETUP bit and RB_XBUS_HOLD bit, to select the bus setup time and bus hold time;
3. Configure the corresponding pin direction of external bus interface, and refer to the description in Section 5.3.1.
4. Drive and point to the address range of 0x80000000 to 80007FFF to read/write data.

## 11.3 Register description

BUS8 peripheral register base address: 0x40001000

Table 11-1 BUS8 register

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_XBUS_CONFIG | 0x10 | External bus configuration register | 0x00 |

External bus configuration register (R8_XBUS_CONFIG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_XBUS_SETUP | RW | External bus data setup time:<br>1: 3 clock cycles;<br>0: 2 clock cycles. | 0 |
| 6 | RB_XBUS_HOLD | RW | External bus data hold time: | 0 |

| | | | | |
|---|---|---|---|---|
| | | | 1: 3 clock cycles;<br>0: 2 clock cycles. | |
| [5:4] | RB_XBUS_WIDTH | RW | External bus RD/WR valid time:<br>00: 3 clock cycles;<br>01: 5 clock cycles;<br>10: 9 clock cycles;<br>11: 16 clock cycles. | 0 |
| [3:2] | RB_XBUS_ADDR_OE | RW | External bus address output enable:<br>00: No bus address output;<br>01: Address lines PA[5:0];<br>10: Address lines PA[9:0];<br>11: Address lines PA[14:0]. | 0 |
| 1 | Reserved | RO | Reserved. | 0 |
| 0 | RB_XBUS_ENABLE | RW | External bus enable:<br>1: Enabled;<br>0: Disabled. | 0 |

# Chapter 12 Digital video port (DVP)

The system provides a Digital Video Port (DVP), which supports obtaining image data streams by using DVP interface timing, supports image data organized in original line and frame formats, such as YUV and RGB, and also supports compressed image data stream in formats such as JPEG. Mainly relies on VSYNC and HSYNC signal synchronization when receiving.

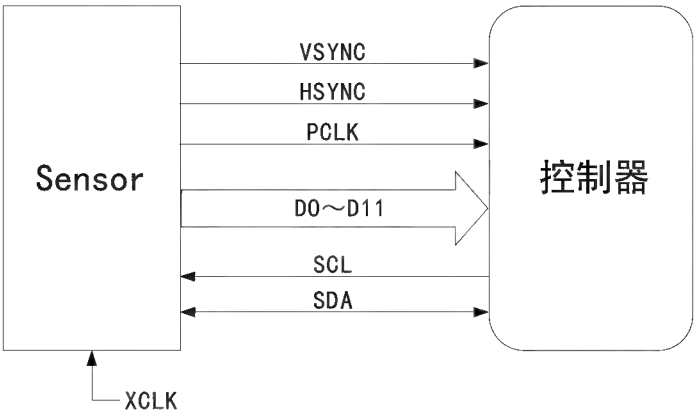
## 12.1 Main features

- Configurable 8/10/12 bit data width mode
- Support the data in formats such as YUV and RGB
- Support the data in JPEG compression format
- Built-in FIFO, support DMA transmission
- Support dual buffering receive/transmit

## 12.2 Functional description

### 12.2.1 Connected to sensor

Figure 12-1 DVP connection




- PCLK (Pixel clk): Pixel clocks. Each clock corresponds to a pixel data (uncompressed). The PCLK clock output by the external DVP sensor generally does not exceed 96MHz.
- HSYNC: Horizontal synchronization signal.
- VSYNC: Vertical synchronization signal.
- DATA: Pixel data or compressed data, with a width of generally 8/10/12 bits.
- XCLK: Reference clock of Sensor, which can be provided by the microcontroller or externally. Usually a crystal oscillator is used.
- I2C interface: Used to read/write the internal registers of sensor and configure parameters. I2C timing communication can be simulated by controlling the ordinary GPIO port.

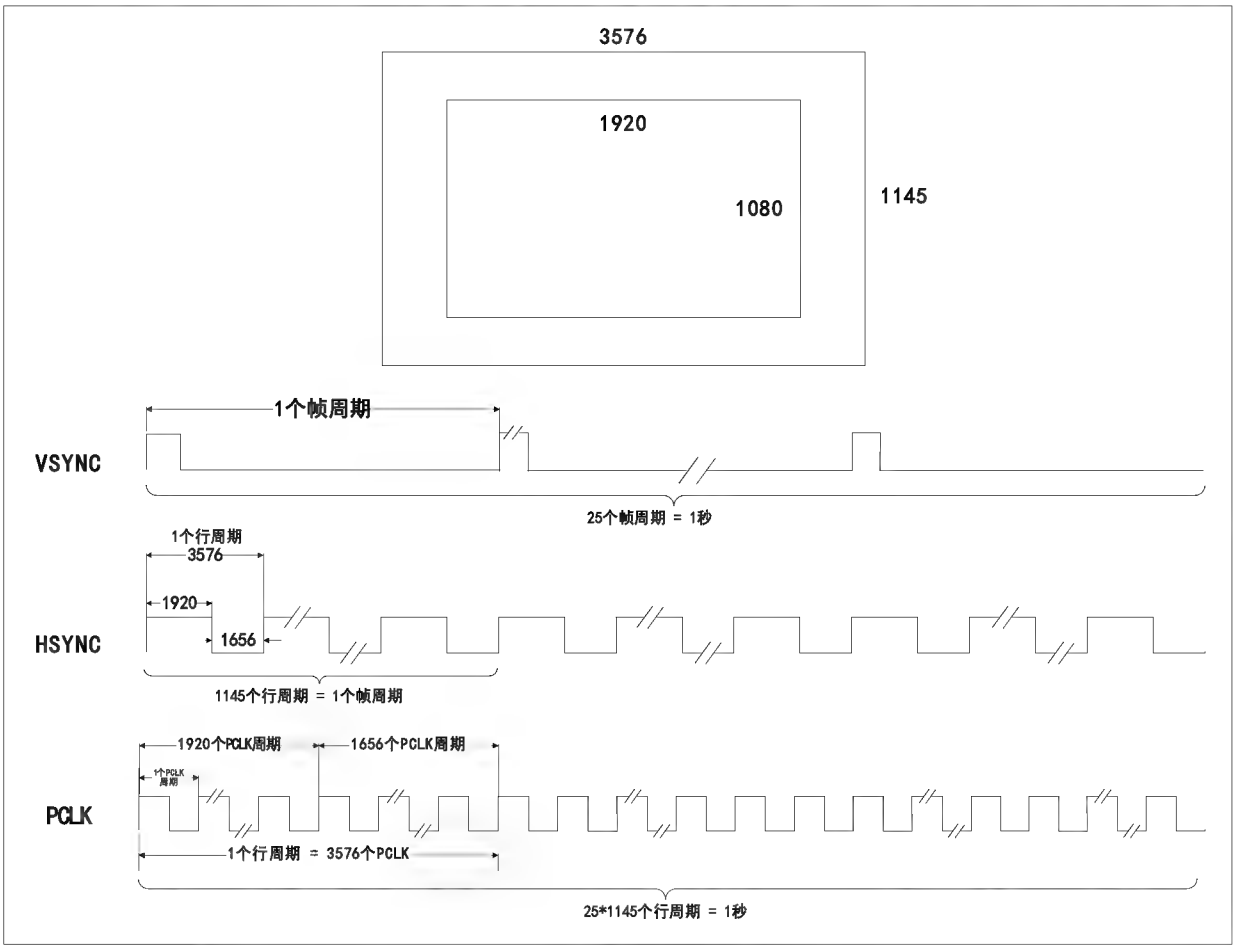
### 12.2.2 Description of DVP timing

The digital signal data stream output by the commonly used DVP sensor module has a relationship with the image size. The following is an example of image data.

As shown in Figure 12-2, a complete image with the size of 3576*1145 is in the sensor. After internal scaling,

the size of the image data finally output from the interface is 1920*1080, and the image refresh rate is 25.

Figure 2-2 Description of timing




PCLK is the transfer time of a pixel, so HSYNC is 3576 times of PCLK. Among 3576 pixels, only 1920 pixels are valid, and the sensor does not transfer data during the time of the remaining 1656 pixels. VSYNC is a frame synchronization signal, so the VSYNC time is 3576*1145 times that of PCLK. Similarly, sensor transfers data only within the time of 1920*1080 valid pixels. If the data transferred by sensor is JPEG compressed data, the HSYNC signal may not be required.

The specific relationship between DVP signal and image data is mainly described in the selected sensor data manual.

### 12.2.3 Description of RGB/YUV/JPEG

- RGB

Three primary colors: red, green, blue

- YUV

Luminance signal Y, chrominance signals U and V, Pb and Pr, Cb and Cr.

- JPEG (Joint Photographic Experts Group)

This is lossy compression, but the lost part is not easily perceived by human vision. It takes advantage of the fact that the human eye is not sensitive to high-frequency information in computer colors. Remove visually redundant information (space redundancy) and remove redundant information of the data itself (structure

redundancy).

## 12.3 Application of digital video port

### 12.3.1 Description of digital video port configuration

(1) RAMX area is used for DVP data storage, and the maximum available space is 96KB (depending on system configuration).

(2) In DVP data reception, each frame of data is alternately stored by BUF0 and BUF1, starting from BUF0. For RGB and YUV data streams, the hardware will reset and select BUF0 to start at the rising edge of each frame signal, and switch to BUF1 after one row of data is stored, to implement alternate storage. For JPEG compressed data, the hardware will set the switching threshold of BUF0 and BUF1 according to the set DMA reception length.

(3) When the data bus width is 10 or 12, the system automatically expands the data to 16 bits without sign before storing after each time a piece of data is received.

(4) The R16_DVP_ROW_NUM and R16_DVP_COL_NUM registers must match the size of the actual image output by the sensor.

(5) In the video stream RGB mode, R16_DVP_COL_NUM represents the number of valid PCLK cycles for a row of data. In the image JPEG mode, R16_DVP_COL_NUM is used to configure the DMA length. In this mode, R16_DVP_ROW_NUM register will not work.

### 12.3.2 Description of digital video port applications

When using the digital video port to receive image data, the DVP control registers must be properly configured to match the mode of image sensor. The specific operation steps are as follows:

a) Clear RB_DVP_ALL_CLR through the R8_DVP_CR1 register, and release the receiving reset.

b) Configure the effective bit width of the received data through R8_DVP_CR0 register.

c) According to the effective image pixels output by the configured image sensor, configure the R16_DVP_ROW_NUM and R16_DVP_COL_NUM registers, to make them match the output of SENSOR.

d) Configure DMA reception address through R32_DVP_DMA_BUF0/1 register.

e) Enable row end interrupt and frame end interrupt through R8_DVP_INT_EN register.

f) Enable DMA through R8_DVP_CR1 register, and enable DVP interface through R8_DVP_CR0 register.

g) Wait for the relevant receive interrupt to be generated, and process the received data in time.

## 12.4 Register description

DVP peripheral register base address: 0x4000E000

Table 12-1 Digital video port registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_DVP_CR0 | 0x00 | DVP control register 0 | 0x00 |
| R8_DVP_CR1 | 0x01 | DVP control register 1 | 0x06 |
| R8_DVP_INT_EN | 0x02 | DVP interrupt enable register | 0x00 |
| R16_DVP_ROW_NUM | 0x04 | Image row number configuration register | 0x0000 |

| | | | |
|---|---|---|---|
| R16_DVP_COL_NUM | 0x06 | Image column number configuration register | 0x0000 |
| R32_DVP_DMA_BUF0 | 0x08 | DVP DMA address 0 register | 0x00000000 |
| R32_DVP_DMA_BUF1 | 0x0C | DVP DMA address 1 register | 0x00000000 |
| R8_DVP_INT_FLAG | 0x10 | DVP interrupt flag register | 0x00 |
| R8_DVP_FIFO_ST | 0x11 | DVP receive FIFO status register | 0x00 |
| R16_DVP_ROW_CNT | 0x14 | DVP row counter | 0x0000 |
| R16_DVP_COL_CNT | 0x16 | DVP column counter | 0x0000 |

DVP control register (R8_DVP_CR0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_DVP_ROW_CM | RW | DVP row count mode control.<br>1: Count at the end of column counting;<br>0: Count on the falling edge of HSYNC. | 0 |
| 6 | RB_DVP_JPEG | RW | JPEG mode enable.<br>1: JPEG compression format;<br>0: Original data format. | 0 |
| [5:4] | RB_DVP_MSK_DAT_MOD | RW | DVP data bit width configuration.<br>00: 8-bit mode;<br>01: 10-bit mode;<br>1x: 12-bit mode. | 0 |
| 3 | RB_DVP_P_POLAR | RW | PCLK polarity configuration.<br>1: Sample data on the falling edge of PCLK;<br>0: Sample data on the rising edge of PCLK. | 0 |
| 2 | RB_DVP_H_POLAR | RW | HSYNC polarity configuration.<br>1: HSYNC low level data valid;<br>0: HSYNC high level data valid. | 0 |
| 1 | RB_DVP_V_POLAR | RW | VSYNC polarity configuration.<br>1: VSYNC high level data valid;<br>0: VSYNC low level data valid. | 0 |
| 0 | RB_DVP_ENABLE | RW | DVP enable.<br>1: DVP enabled; 0: DVP disabled. | 0 |

DVP control register (R8_DVP_CR1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:4] | Reserved | RO | Reserved. | - |
| 3 | RB_DVP_BUF_TOG | RWT | Buffer address flag. Toggled by hardware control. Toggled when set to 1 by software, and it is invalid when writing 0.<br>1: Data stored in the reception address 1;<br>0: Data stored in the reception address 0. | 0 |
| 2 | RB_DVP_RCV_CLR | RW | Receive logic reset control.<br>1: Reset the receive logic circuit;<br>0: Cancel reset. | 1 |

| | | | | |
|---|---|---|---|---|
| 1 | RB_DVP_ALL_CLR | RW | Flag and FIFO clear control. Write 1 or write 0 by software:<br>1: Reset flag and FIFO;<br>0: Cancel reset. | 1 |
| 0 | RB_DVP_DMA_ENABLE | RW | DMA enable:<br>1: DMA enabled; 0: DMA disabled. | 0 |

DVP interrupt enable register (R8_DVP_INT_EN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RO | Reserved. | - |
| 4 | RB_DVP_IE_STP_FRM | RW | Frame stop interrupt enable. Interrupt is generated on the rising edge of VSYNC.<br>1: Frame stop interrupt enabled;<br>0: Frame stop interrupt disabled. | 0 |
| 3 | RB_DVP_IE_FIFO_OV | RW | Receive FIFO overflow interrupt enable.<br>1: FIFO overflow interrupt enabled;<br>0: FIFO overflow interrupt disabled. | 0 |
| 2 | RB_DVP_IE_FRM_DONE | RW | Frame reception complete interrupt enable. (The counter reaches the RAW/COL_NUM configuration value)<br>1: Frame reception complete interrupt enabled;<br>0: Frame reception complete interrupt disabled. | 0 |
| 1 | RB_DVP_IE_ROW_DONE | RW | Row done interrupt enable. (The counter reaches the COL_NUM configuration value or falling edge of HSYNC)<br>1: Row down interrupt enabled;<br>0: Row down interrupt disabled. | 0 |
| 0 | RB_DVP_IE_STR_FRM | RW | New frame start interrupt enable. Interrupt is generated on the falling edge of VSYNC, indicating that a new frame starts and data is coming.<br>1: New frame start interrupt enabled;<br>0: New frame start interrupt disabled; | 0 |

DVP image valid row number configuration register (R16_DVP_ROW_NUM)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | RB_DVP_ROW_NUM | RW | In RGB mode, it indicates the number of rows contained in a frame of image data. In JPEG mode, this register has no practical meaning. | 0 |

DVP image valid column number configuration register (R16_DVP_COL_NUM)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | RB_DVP_COL_NUM | RW | In RGB mode, it indicates the number of PCLK cycles contained in a row of data. In JPEG mode, it is used to configure the DMA reception length. | 0 |

DVP DMA reception address 0 register (R32_DVP_DMA_BUF0)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | - |
| [16:0] | RB_DVP_DMA_BUF0 | RW | DMA reception address 0. The lower 4 bits are fixed to 0 (16 bytes are aligned). | 0 |

*Note: DMA addresses RAMX area.*

DVP DMA reception address 1 register (R32_DVP_DMA_BUF1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | - |
| [16:0] | RB_DVP_DMA_BUF1 | RW | DMA reception address 1. The lower 4 bits are fixed to 0 (16 bytes are aligned). | 0 |

*Note: DMA addresses RAMX area.*

DVP interrupt flag register (R8_DVP_INT_FLAG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:5] | Reserved | RO | Reserved. | - |
| 4 | RB_DVP_IF_STP_FRM | RW | Frame stop interrupt flag, active high, cleared by writing 1. | 0 |
| 3 | RB_DVP_IF_FIFO_OV | RW | Receive FIFO overflow interrupt flag, active high, cleared by writing 1. | 0 |
| 2 | RB_DVP_IF_FRM_DONE | RW | Frame done interrupt flag, active high, cleared by writing 1. | 0 |
| 1 | RB_DVP_IF_ROW_DONE | RW | Row done interrupt flag, active high, cleared by writing 1. | 0 |
| 0 | RB_DVP_IF_STR_FRM | RW | Frame start interrupt flag, active high, cleared by writing 1. | 0 |

DVP receive FIFO status register (R8_DVP_FIFO_ST)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | Reserved | RO | Reserved. | 0 |
| [6:4] | RB_DVP_FIFO_CNT | RO | FIFO counter. | 0 |
| 3 | Reserved | RO | Reserved. | 0 |

| | | | | |
|---|---|---|---|---|
| 2 | RB_DVP_FIFO_OV | RO | FIFO overflow status.<br>1: FIFO overflow; 0: No FIFO overflow. | 0 |
| 1 | RB_DVP_FIFO_FULL | RO | FIFO full status.<br>1: Buffer full; 0: FIFO not full. | 0 |
| 0 | RB_DVP_FIFO_RDY | RO | FIFO ready status.<br>1: There is data in FIFO;<br>1: No data in FIFO. | 0 |

DVP receive image row count register (R16_DVP_ROW_CNT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | RB_DVP_ROW_CNT | RO | In actual reception, the number of rows contained in a frame of image data. This register is updated at the end of frame.<br>In the JPEG format, the value of this register has no meaning. | 0 |

DVP receive image column count register (R16_DVP_COL_CNT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | RB_DVP_COL_CNT | RO | In the original data format:<br>It indicates the number of PCLKs contained in a row of image data during actual reception. This register is updated at the end of row.<br>In JPEG format:<br>It indicates the number of PCLKs contained in a frame of data packet during actual reception. This register is updated at the end of frame. | 0 |

# Chapter 13 USB2.0 controller and transceiver (USBHS)

USB2.0 controller has the dual roles of USB host controller and USB device controller, and has an embedded USB-PHY unit. When used as a host controller, it can support low-speed, full-speed and high-speed USB devices. When used as a device controller, it can be flexibly set to low-speed, full-speed or high-speed mode, to adapt to various applications.

## 13.1 Main features

- Support USB 2.0, USB 1.1, USB 1.0 Specifications
- Support USB Host function and USB Device function
- Both the host and the device support control transfer, bulk transfer, interrupt transfer, isochronous transfer
- Host supports high-speed HUB
- Can be configured as high-speed, full-speed or low-speed device mode in device mode
- Provide 8 groups of upper and lower transfer channels in device mode, and support to configure16 endpoint numbers
- Support to directly access the data of each endpoint buffer by DMA
- Support bus reset, suspend, remote wake-up and resume functions
- Except device endpoint0, all other endpoints support the data packets up to 1024 bytes, and dual buffering is available

## 13.2 Functional description

## 13.3 USB device mode configuration

1. Set the R8_USB_CTRL register, set the RB_USB_MODE bit to 0, to configure USB device mode;
2. Set the R8_USB_CTRL register, set RB_USB_RESET_SIE and RB_USB_CLR_ALL bits to 0, set RB_USB_INT_BUSY and RB_USB_DMA_EN bits to 1, configure RB_USB_SPTP_MASK to select the speed of USB device. If it is set as a high-speed device, but the current host is at full speed, the controller automatically slows down and switches to full speed, and the actual communication speed can be queried by the R8_USB_SPD_TYPE register.
3. Clear R8_USB_DEV_AD and R8_USB_INT_FG, optional operation, to enable the required interrupt, and write into the R8_USB_INT_EN register;
4. Configure the device endpoint data receive/transmit buffer mode registers (R8_UEP4_1_MOD / R8_UEP2_3_MOD / R8_UEP5_6_MOD / R8_UEP7_MOD), and configure the receive/transmit control registers (R8_UEPn_TX_CTRL / R8_UEPn_RX_CTRL);
5. Set the endpoint maximum packet reception length register (R16_UEPn_MAX_LEN) and the endpoint data receive/transmit start address (R32_UEPn_RX_DMA / R32_UEPn_TX_DMA);
6. Set the RB_DEV_PU_EN bit in the R8_USB_CTRL register to 1, to enable USB device function.

## 13.4 USB host mode configuration

1. Set the R8_USB_CTRL register, set the RB_USB_MODE bit to 1, to configure USB host mode;
2. Set the R8_USB_CTRL register, clear RB_USB_RESET_SIE and RB_USB_CLR_ALL bits to 0, set RB_USB_INT_BUSY and RB_USB_DMA_EN bits to 1, configure RB_USB_SPTP_MASK to select the

speed of USB host. If it is set as a high-speed host, but the device currently connected is at full speed, the controller automatically slows down and switches to full speed, and actual communication speed can be queried in the R8_USB_SPD_TYPE register.

3. Clear R8_USB_DEV_AD and R8_USB_INT_FG, optional operation, enable the required interrupt, and write into the R8_USB_INT_EN register;
4. Configure the host endpoint data receive/transmit buffer mode register (R8_UH_EP_MOD) to enable receive/transmit channel, and configure the receive/transmit control registers (R8_UH_RX_CTRL / R8_UH_TX_CTRL);
5. Set the endpoint maximum packet reception length register (R16_UH_MAX_LEN) and the host endpoint data receive/transmit start address registers (R32_UEPn_RX_DMA / R32_UEPn_TX_DMA);
6. Set the RB_UH_AUTOSOF_EN bit in the R8_UHOST_CTRL register to 1, to enable port to transmit SOF packet automatically.

## 13.5 Register description

The USB2.0 master-slave controller (built-in PHY) can be flexibly configured as a host or device. The related registers are divided into 3 parts, some of which are multiplexed in the host and device modes.

- USB global registers
- USB device controller registers
- USB host controller registers

USBHS register physical base address: 0x40009000

### 13.5.1 Global register description

Table 13-1 USBHS global registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_USB_CTRL | 0x00 | USB control register | 0x06 |
| R8_USB_INT_EN | 0x02 | USB interrupt enable register | 0x00 |
| R8_USB_DEV_AD | 0x03 | USB address register | 0x00 |
| R16_USB_FRAME_NO | 0x04 | USBHS frame number register | 0x0000 |
| R8_USB_SUSPEND | 0x06 | USB suspend control register | 0x00 |
| R8_USB_SPD_TYPE | 0x08 | USB current speed type register | 0x00 |
| R8_USB_MIS_ST | 0x09 | USB miscellaneous status register | 0x20 |
| R8_USB_INT_FG | 0x0A | USB interrupt flag register | 0x00 |
| R8_USB_INT_ST | 0x0B | USB interrupt status register | 0xXX |
| R16_USB_RX_LEN | 0x0C | USB reception length register | 0xXXXX |

USB control register (R8_USB_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_USB_MODE | RW | USB working mode selection bit:<br>0: Device mode;<br>1: Host mode. | 0 |
| [6:5] | RB_USB_SPTP_MASK | RW | USB bus signal transfer rate selection bit:<br>00: Full speed; 01: High speed; 10: Low speed. | 0 |

| | | | | |
|---|---|---|---|---|
| 4 | RB_DEV_PU_EN | RW | In device mode, USB device enable and internal pull-up resistor control bit:<br>1: USB device transfer enabled and internal pull-up resistor enabled;<br>0: Disabled. | 0 |
| 3 | RB_USB_INT_BUSY | RW | Auto pause enable bit before the USB transfer complete interrupt flag is not cleared:<br>1: Automatically pause before the interrupt flag RB_USB_IF_TRANSFER is not cleared. Automatically respond busy NAK in device mode. Automatically pause subsequent transfer in host mode;<br>0: No pause. | 0 |
| 2 | RB_USB_RESET_SIE | RW | Software reset control bit of USB protocol processor:<br>1: Forcibly reset the USB protocol processor (SIE). Cleared by software;<br>0: No reset. | 1 |
| 1 | RB_USB_CLR_ALL | RW | 1: Empty USB interrupt flag and FIFO. Cleared by software;<br>0: Not empty. | 1 |
| 0 | RB_USB_DMA_EN | RW | USB DMA and DMA interrupt enable bit:<br>1: DMA and DMA interrupt enabled;<br>0: DMA disabled. | 0 |

USB interrupt enable register (R8_USB_INT_EN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_USB_IE_DEV_NAK | RW | In USB device mode, receive NAK interrupt enable:<br>1: Interrupt enabled;<br>0: Interrupt disabled. | 0 |
| 6 | RB_USB_IE_ISOACT | RW | Synchronous transfer receive control token packet interrupt enable:<br>1: Interrupt enabled;<br>0: Interrupt disabled. | 0 |
| 5 | RB_USB_IE_SETUPACT | RW | SETUP transaction complete interrupt:<br>1: Interrupt enabled;<br>0: Interrupt disabled. | 0 |
| 4 | RB_USB_IE_FIFOOV | RW | Internal FIFO overflow interrupt enable:<br>1: Interrupt enabled;<br>0: Interrupt disabled. | 0 |
| 3 | RB_USB_IE_SOF | RW | In USB host mode, SOF timing interrupt enable:<br>1: Interrupt enabled;<br>0: Interrupt disabled.<br>In USB device mode, SOF packet interrupt is received: | 0 |
| 2 | RB_USB_IE_SUSPEND | RW | USB bus suspend or wake-up event interrupt enable: | 0 |

| | | | | |
|---|---|---|---|---|
| | | | 1: Interrupt enabled;<br>0: Interrupt disabled. | |
| 1 | RB_USB_IE_TRANS | RW | USB transfer complete interrupt enable:<br>1: Interrupt enabled;<br>0: Interrupt disabled. | 0 |
| 0 | RB_USB_IE_DETECT | RW | In USB host mode, USB device connection or disconnection event interrupt enable:<br>1: Interrupt enabled;<br>0: Interrupt disabled. | 0 |
| 0 | RB_USB_IE_BUSRST | RW | In USB device mode, USB bus reset event interrupt enable:<br>1: Interrupt enabled;<br>0: Interrupt disabled. | 0 |

USB device address register (R8_USB_DEV_AD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | Reserved | RO | Reserved. | 0 |
| [6:0] | USB_ADDR_MASK | RW | In host mode, it is the address of the USB device (HUB) being operated. In device mode, it is the address of the USB device. | 0 |

USB frame number register (R16_USB_FRAME_NO)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | USB_FRAME_NO | RO | Frame number. It indicates the frame number of the SOF packet to be transmitted in host mode, and it indicates the frame number of the SOF packet currently received in device mode. The lower 11 bits are the valid frame number, and the higher 3 bits are the frame number of high-speed mode. | 0 |

*USB_FRAME_NO is a 16-bit register. The lower 11 bits represent the SOF packet frame number, and the higher 3 bits represents the current frame number, which can be used for interrupt, synchronous/real-time transfer when operating high-speed HUB.*

USB suspend register (R8_USB_SUSPEND)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:2] | Reserved | RO | Reserved. | 0 |
| 1 | RB_DEV_WAKEUP | RW | Remote wake-up control bit:<br>1: Wake up the host remotely;<br>0: No action. | 0 |
| 0 | Reserved | RO | Reserved. | 0 |

*Note: When the device wakes up the host remotely, pull the bUS_RESUME bit up and then down.*

USB speed type register (R8_USB_SPD_TYPE)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:2] | Reserved | RO | Reserved. | 0 |
| [1:0] | RB_USBSPEED_MASK | RO | Actual transfer speed.<br>In host mode, it indicates the speed type of the currently connected device. In device mode, it indicates the speed type of the current device;<br>00: Full speed; 01: High speed; 10: Low speed. | 0 |

*Note: Different from RB_USB_SPTP_MASK in the R8_USB_CTRL register, RB_USB_SPTP_MASK represents the highest speed expected. Assuming that in the device mode, set RB_USB_SPTP_MASK to high speed. When the device is connected to a host at full speed, the actual speed type is full speed and it can be known by querying the RB_USBSPEED_MASK register. In host mode, set RB_USB_SPTP_MASK to high speed. When a device at full speed is connected, the actual communication speed is full speed and it can be known by querying the RB_USBSPEED_MAS register.*

USB miscellaneous status register (R8_USB_MIS_ST)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_USB_SOF_PRES | RO | SOF packet presage status bit in USB host mode:<br>1: SOF packet will be sent, and it will be automatically delayed if there are other USB data packets;<br>0: No SOF packet is sent. | X |
| 6 | RB_USB_SOF_ACT | RO | SOF packet transfer status bit in USB host mode:<br>1: SOF packet is being sent out;<br>0: Transmission completed, or idle. | X |
| 5 | RB_USB_SIE_FREE | RO | Free status bit of USB protocol processor:<br>1: Protocol processor free;<br>0: Busy; USB transmission is in progress. | 1 |
| 4 | RB_USB_FIFO_RDY | RO | USB receive FIFO data ready status bit:<br>1: Receive FIFO not empty;<br>0: Receive FIFO empty. | 0 |
| 3 | RB_USBBUS_RESET | RO | USB bus reset status bit:<br>1: The USB bus is reset currently;<br>0: The USB bus is not reset currently. | 0 |
| 2 | RB_USBBUS_SUSPEND | RO | USB suspend status bit:<br>1: USB bus suspended;<br>0: USB bus not suspended. | 0 |
| 1 | RB_USB_ATTACH | RO | USB device connection status bit of the port in the USB host mode:<br>1: The port has been connected to a USB device;<br>0: No USB device is connected to the port. | 0 |
| 0 | RB_HOST_SPLIT_EN | RO | In USB host mode, the SPLIT packet transmit enable:<br>1: SPLIT packet transmission enabled;<br>0: Transmission disabled. | 0 |

USB interrupt flag register (R8_USB_INT_FG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | Reserved | RO | Reserved. | 0 |
| 6 | RB_USB_IF_ISOACT | RW1 | Synchronous transfer receive control token packet interrupt flag. Cleared by writing 1.<br>1: Start transmitting/receiving data trigger;<br>0: No event. | 0 |
| 5 | RB_USB_IF_SETUOACT | RW1 | SETUP transaction complete interrupt flag. Clear by writing 1.<br>1: SETUP transaction completed;<br>0: No event. | 0 |
| 4 | RB_USB_IF_FIFOOV | RW1 | USB FIFO overflow interrupt flag. Cleared by writing 1.<br>1: FIFO overflow trigger;<br>0: No event. | 0 |
| 3 | RB_USB_IF_HST_SOF | RW1 | SOF timing interrupt flag bit in USB host mode. Cleared by writing 1.<br>1: SOF transmission completion trigger;<br>0: No event. | 0 |
| 2 | RB_USB_IF_SUSPEND | RW1 | USB bus suspend or wake-up event interrupt flag. Cleared by writing 1.<br>1: USB suspend event or wake-up event trigger;<br>0: No event. | 0 |
| 1 | RB_USB_IF_TRANSFER | RW1 | USB transmission complete interrupt flag. Cleared by writing 1.<br>1: A USB transmission completion trigger;<br>0: No event. | 0 |
| 0 | RB_USB_IF_DETECT | RW1 | In USB host mode, USB device connection or disconnection event interrupt flag. Cleared by writing 1:<br>1: USB device connection or disconnection trigger detected;<br>0: No event. | 0 |
| 0 | RB_USB_IF_BUSRST | RW1 | USB bus reset event interrupt flag in USB device mode. Cleared by writing 1.<br>1: USB bus reset event trigger;<br>0: No event. | 0 |

USB interrupt status register (R8_USB_INT_ST)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_USB_ST_NAK | RO | In USB device mode, NAK response status bit:<br>1: Respond NAK during current USB transfer;<br>0: No NAK. | 0 |
| 6 | RB_USB_ST_TOGOK | RO | Current USB transmit DATA0/1 synchronization flag matching status bit:<br>1: Synchronous;<br>0: Asynchronous. | 0 |

| | | | | |
|---|---|---|---|---|
| [5:4] | RB_DEV_TOKEN_MASK | RO | In device mode, the token PID of the current USB transfer transaction. | X |
| [3:0] | RB_DEV_ENDP_MASK | RO | In device mode, the endpoint number of the current USB transfer transaction. | X |
| [3:0] | RB_HOST_RES_MASK | RO | In the host mode, the response PID mask of the current USB transfer transaction.<br>0000: Device has no response, or timeout;<br>Other values: Response PID. | X |

RB_DEV_TOKEN_MASK is used to identify the token PID of the current USB transfer transaction in USB device mode: 00 represents OUT packet; 01 represents SOF packet; 10 represents IN packet;
RB_HOST_RES_MASK is only valid in host mode. In host mode, if the host sends an OUT/SETUP token packet, the PID is the handshake packet ACK/NAK/STALL/NYET, or the device has no response/timeout. If the host sends an IN token packet, the PID is the PID of the data packet (DATA0/DATA1/DATA2/MDATA) or the handshake packet PID.

USB reception length register (R16_USB_RX_LEN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | USB_RX_LEN | RO | Current count of data received by the USB endpoint. The lower 11 bits are valid, and the higher 5 bits are fixed to 0. | X |

### 13.5.2 Device register description

The USB device controller is equipped with 8 sets of bidirectional endpoints configuration registers (endpoint0 to endpoint7). The configuration of endpoint0 to endpoint7 can map the configuration of endpoint8 to endpoint15. The maximum data packet length of all endpoints except endpoint0 is 1024 bytes, and the maximum data packet length of endpoint 0 is 64 bytes.

- Endpoint0 is the default endpoint, which supports control transfer. The transmission and reception share a 64-byte data buffer.
- Endpoint1 to endpoint15 can be configured with independent reception and transmission buffers which are up to 1024 bytes or data buffers which are up to dual 1024 bytes, and they support bulk transfer, interrupt transfer, and isochronous transfer.

Endpoint0 has an independent DMA address which is shared for reception and transmission, and endpoint1 to endpoint7 (endpoint8 to endpoint15) are provided with one DMA address for transmission and one DMA address for reception. The mode of data buffer can be set to dual buffer or single buffer through R8_UEPn_*_MODE register. If the dual buffer mode is selected, the endpoint can only be configured with unidirectional transfer.

Each group of endpoints is equipped with receive/transmit control registers (UEPn_TX_CTRL and UEPn_RX_CTRL) and transmission length registers, UEPn_T_LEN and UEPn_*_DMA (n=0~7), which are used to set the synchronization trigger bit of endpoint, the response to OUT transactions and IN transactions and the length of data to be sent and other parameters.

As the necessary USB bus pull-up resistor of USB device, it can be set to be enabled by software at any time. When RB_UC_DEV_PU_EN in the R8_USB_CTRL is set to 1, hardware sets it according to the speed of bUC_SPEED_TYPE, internally connect the pull-up resistor with the DP/DM pin of the USB bus and enable

the USB device function.

When a USB bus reset, or USB bus suspend or wakeup event is detected, or when the USB successfully processes data transmission or reception, the USB protocol processor sets corresponding interrupt flag. If the interrupt enable is switched on, the corresponding interrupt request is also generated. The application program can directly query or query and analyze the interrupt flag register (R8_USB_INT_FG) in the USB interrupt service program, and perform corresponding processing according to RB_USB_IF_BUSRST and RB_USB_IF_SUSPEND. In addition, if RB_USB_IF_TRANSFER is valid, it is required to continue to analyze the USB interrupt status register (R8_USB_INT_ST), and perform the corresponding processing according to the current endpoint number (RB_DEV_ENDP_MASK) and the current transaction token PID identifier (RB_DEV_TOKEN_MASK). If the synchronization trigger bit (RB_UEP_R_TOG_MASK) of OUT transaction of each endpoint is set in advance, you can judge whether the synchronization trigger bit of the data packet received matches the synchronization trigger bit of the endpoint through RB_USB_ST_TOGOK. If the data is synchronized, the data is valid. If the data is not synchronized, the data should be discarded. Every time the USB transmit/receive interrupt is processed, the synchronization trigger bit of corresponding endpoint should be modified correctly to synchronize the data packet sent next time and detect whether the data packet received next time is synchronized. In addition, RB_UEP_T_AUTOTOG/RB_UEP_R_AUTOTOG can be set to automatically toggle the corresponding synchronization trigger bit after successful reception or transmission.

The data to be sent by each endpoint is in its own buffer, and the length of the data to be sent is independently set in R16_UEPn_T_LEN. The data received by each endpoint is in its own buffer, but the length of the data received is in the USB length reception register (R16_USB_RX_LEN), and it can be distinguished according to the current endpoint number when the USB is receiving an interrupt. The maximum packet length that can be received by each endpoint needs to be written into the R16_UEPn_MAX_LEN register in advance.

Table 13-2 USB device registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_UEP4_1_MOD | 0x10 | Endpoint 1(9)/4(8/12) mode control register | 0x00 |
| R8_UEP2_3_MOD | 0x11 | Endpoint 2(10)/3(11) mode control register | 0x00 |
| R8_UEP5_6_MOD | 0x12 | Endpoint 5(13)/6(14) mode control register | 0x00 |
| R8_UEP7_MOD | 0x13 | Endpoint 7 (15) mode control register | 0x00 |
| R32_UEP0_RT_DMA | 0x14 | Start address of endpoint0 buffer | 0x000XXXXX |
| R32_UEP1_RX_DMA | 0x18 | Start address of endpoint 1(9) receive buffer | 0x000XXXXX |
| R32_UEP2_RX_DMA | 0x1C | Start address of endpoint 2(10) receive buffer | 0x000XXXXX |
| R32_UEP3_RX_DMA | 0x20 | Start address of endpoint 3(11) receive buffer | 0x000XXXXX |
| R32_UEP4_RX_DMA | 0x24 | Start address of endpoint 4(8/12) receive buffer | 0x000XXXXX |
| R32_UEP5_RX_DMA | 0x28 | Start address of endpoint 5(13) receive buffer | 0x000XXXXX |
| R32_UEP6_RX_DMA | 0x2C | Start address of endpoint 6(14) receive buffer | 0x000XXXXX |
| R32_UEP7_RX_DMA | 0x30 | Start address of endpoint 7(15) receive buffer | 0x000XXXXX |
| R32_UEP1_TX_DMA | 0x34 | Start address of endpoint 1(9) transmit buffer | 0x000XXXXX |
| R32_UEP2_TX_DMA | 0x38 | Start address of endpoint 2(10) transmit buffer | 0x000XXXXX |
| R32_UEP3_TX_DMA | 0x3C | Start address of endpoint 3(11) transmit buffer | 0x000XXXXX |
| R32_UEP4_TX_DMA | 0x40 | Start address of endpoint 4(8/12) transmit buffer | 0x000XXXXX |
| R32_UEP5_TX_DMA | 0x44 | Start address of endpoint 5(13) transmit buffer | 0x000XXXXX |

| | | | |
|---|---|---|---|
| R32_UEP6_TX_DMA | 0x48 | Start address of endpoint 6(14) transmit buffer | 0x000XXXXX |
| R32_UEP7_TX_DMA | 0x4C | Start address of endpoint 7(15) transmit buffer | 0x000XXXXX |
| R16_UEP0_MAX_LEN | 0x50 | Endpoint 0 receive maximum length packet register | 0xXXXX |
| R16_UEP1_MAX_LEN | 0x54 | Endpoint 1(9) receive maximum length packet register | 0xXXXX |
| R16_UEP2_MAX_LEN | 0x58 | Endpoint 2(10) receive maximum length packet register | 0xXXXX |
| R16_UEP3_MAX_LEN | 0x5C | Endpoint 3(11) receive maximum length packet register | 0xXXXX |
| R16_UEP4_MAX_LEN | 0x60 | Endpoint 4(8/12) receive maximum length packet register | 0xXXXX |
| R16_UEP5_MAX_LEN | 0x64 | Endpoint 5(13) receiving maximum length packet register | 0xXXXX |
| R16_UEP6_MAX_LEN | 0x68 | Endpoint 6(14) receive maximum length packet register | 0xXXXX |
| R16_UEP7_MAX_LEN | 0x6C | Endpoint 7(15) receive maximum length packet register | 0xXXXX |
| R16_UEP0_T_LEN | 0x70 | Endpoint 0 transmission length register | 0xXXXX |
| R8_UEP0_TX_CTRL | 0x72 | Endpoint 0 transmit control register | 0x00 |
| R8_UEP0_RX_CTRL | 0x73 | Endpoint 0 receive control register | 0x00 |
| R16_UEP1_T_LEN | 0x74 | Endpoint 1(9) transmission length register | 0xXXXX |
| R8_UEP1_TX_CTRL | 0x76 | Endpoint 1(9) transmit control register | 0x00 |
| R8_UEP1_RX_CTRL | 0x77 | Endpoint 1(9) receive control register | 0x00 |
| R16_UEP2_T_LEN | 0x78 | Endpoint 2 (10) transmission length register | 0xXXXX |
| R8_UEP2_TX_CTRL | 0x7A | Endpoint 2(10) transmit control register | 0x00 |
| R8_UEP2_RX_CTRL | 0x7B | Endpoint 2(10) receive control register | 0x00 |
| R16_UEP3_T_LEN | 0x7C | Endpoint 3(11) transmission length register | 0xXXXX |
| R8_UEP3_TX_CTRL | 0x7E | Endpoint 3(11) transmit control register | 0x00 |
| R8_UEP3_RX_CTRL | 0x7F | Endpoint 3(11) receive control register | 0x00 |
| R16_UEP4_T_LEN | 0x80 | Endpoint 4(8/12) transmission length register | 0xXXXX |
| R8_UEP4_TX_CTRL | 0x82 | Endpoint 4(8/12) transmit control register | 0x00 |
| R8_UEP4_RX_CTRL | 0x83 | Endpoint 4(8/12) receive control register | 0x00 |
| R16_UEP5_T_LEN | 0x84 | Endpoint 5(13) transmission length register | 0xXXXX |
| R8_UEP5_TX_CTRL | 0x86 | Endpoint 5(13) transmit control register | 0x00 |
| R8_UEP5_RX_CTRL | 0x87 | Endpoint 5(13) receive control register | 0x00 |
| R16_UEP6_T_LEN | 0x88 | Endpoint 6(14) transmission length register | 0xXXXX |
| R8_UEP6_TX_CTRL | 0x8A | Endpoint 6(14) transmit control register | 0x00 |
| R8_UEP6_RX_CTRL | 0x8B | Endpoint 6(14) receive control register | 0x00 |
| R16_UEP7_T_LEN | 0x8C | Endpoint 7(15) transmission length register | 0xXXXX |
| R8_UEP7_TX_CTRL | 0x8E | Endpoint 7(15) transmit control register | 0x00 |
| R8_UEP7_RX_CTRL | 0x8F | Endpoint 7(15) receive control register | 0x00 |

USB endpoint 1(9)/4(8/12) mode control register (R8_UEP4_1_MOD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_UEP1_RX_EN | RW | 1: Endpoint 1(9) reception enabled (OUT);<br>0: Endpoint 1(9) reception disabled. | 0 |
| 6 | RB_UEP1_TX_EN | RW | 1: Endpoint 1(9) transmission enabled (IN);<br>0: Endpoint 1(9) transmission disabled. | 0 |
| 5 | Reserved | RO | Reserved. | 0 |
| 4 | RB_UEP1_BUF_MOD | RW | Endpoint 1(9) data buffer mode control bit.<br>*Note: When this bit is 1, UEP1_RX_EN and UEP1_TX_EN cannot be 1 at the same time.* | 0 |
| 3 | RB_UEP4_RX_EN | RW | 1: Endpoint 4 (8/12) reception enabled (OUT);<br>0: Endpoint 4 (8/12) reception disabled. | 0 |
| 2 | RB_UEP4_TX_EN | RW | 1: Endpoint 4 (8/12) transmission enabled (IN);<br>0: Endpoint 4 (8/12) transmission disabled. | 0 |
| 1 | Reserved | RO | Reserved. | 0 |
| 0 | RB_UEP4_BUF_MOD | RW | Endpoint 4 (8/12) data buffer mode control bit.<br>*Note: When this bit is 1, UEP4_RX_EN and UEP4_TX_EN cannot be 1 at the same time.* | 0 |

*Note: Endpoint1 configuration option maps to endpoint9. Endpoint4 configuration option maps to endpoint8 or endpoint12.*

Endpoint 2(10)/3(11) mode control register (R8_UEP2_3_MOD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_UEP3_RX_EN | RW | 1: Endpoint 3(11) reception enabled (OUT);<br>0: Endpoint 3(11) reception disabled. | 0 |
| 6 | RB_UEP3_TX_EN | RW | 1: Endpoint 3(11) transmission enabled (IN);<br>0: Endpoint 3(11) transmission disabled. | 0 |
| 5 | Reserved | RO | Reserved. | 0 |
| 4 | RB_UEP3_BUF_MOD | RW | Endpoint 3(11) data buffer mode control bit.<br>*Note: When this bit is 1, UEP3_RX_EN and UEP3_TX_EN cannot be 1 at the same time.* | 0 |
| 3 | RB_UEP2_RX_EN | RW | 1: Endpoint 2(10) reception enabled (OUT);<br>0: Endpoint 2(10) reception disabled. | 0 |
| 2 | RB_UEP2_TX_EN | RW | 1: Endpoint 2 (10) transmission enabled (IN);<br>0: Endpoint 2(10) transmission disabled. | 0 |
| 1 | Reserved | RO | Reserved. | 0 |
| 0 | RB_UEP2_BUF_MOD | RW | Endpoint 2(10) data buffer mode control bit.<br>*Note: When this bit is 1, UEP2_RX_EN and UEP2_TX_EN cannot be 1 at the same time.* | 0 |

*Note: Endpoint3 configuration option maps to endpoint11. Endpoint2 configuration option maps to endpoint10.*

Endpoint 5(13)/6(14) mode control register (R8_UEP5_6_MOD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_UEP6_RX_EN | RW | 1: Endpoint 6(14) reception enabled (OUT);<br>0: Endpoint 6(14) reception disabled. | 0 |
| 6 | RB_UEP6_TX_EN | RW | 1: Endpoint 6(14) transmission enabled (IN);<br>0: Endpoint 6(14) transmission disabled. | 0 |
| 5 | Reserved | RO | Reserved. | 0 |
| 4 | RB_UEP6_BUF_MOD | RW | Endpoint 6(14) data buffer mode control bit.<br>*Note: When this bit is 1, UEP6_RX_EN and UEP6_TX_EN cannot be 1 at the same time.* | 0 |
| 3 | RB_UEP5_RX_EN | RW | 1: Endpoint 5(13) reception enabled (OUT);<br>0: Endpoint 5(13) reception disabled. | 0 |
| 2 | RB_UEP5_TX_EN | RW | 1: Endpoint 5(13) transmission enabled (IN);<br>0: Endpoint 5(13) transmission disabled. | 0 |
| 1 | Reserved | RO | Reserved. | 0 |
| 0 | RB_UEP5_BUF_MOD | RW | Endpoint 5(13) data buffer mode control bit.<br>*Note: When this bit is 1, UEP5_RX_EN and UEP5_TX_EN cannot be 1 at the same time.* | 0 |

*Note: Endpoint5 configuration options map to endpoint13. Endpoint6 configuration options map to endpoint14.*

Endpoint 7 (15) mode control register (R8_UEP7_MOD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:4] | Reserved | RO | Reserved. | 0 |
| 3 | RB_UEP7_RX_EN | RW | 1: Endpoint 7 (15) reception enabled (OUT);<br>0: Endpoint 7(15) reception disabled. | 0 |
| 2 | RB_UEP7_TX_EN | RW | 1: Endpoint 7(15) transmission enabled (IN);<br>0: Endpoint 7(15) transmission disabled. | 0 |
| 1 | Reserved | RO | Reserved. | 0 |
| 0 | RB_UEP7_BUF_MOD | RW | Endpoint 7(15) data buffer mode control bit.<br>*Note: When this bit is 1, UEP7_RX_EN and UEP7_TX_EN cannot be 1 at the same time.* | 0 |

*Note: Endpoint7 configuration option maps to endpoint15.*

The data buffer modes of USB endpoint1 to endpoint15 are controlled by a combination of RB_UEPn_RX_EN, RB_UEPn_TX_EN and RB_UEPn_BUF_MOD respectively, refer to Table 12-4 for details.

Table 12-4 Endpointn buffer modes (n=1-7(8-15))

| RB_UEPn_RX_EN | RB_UEPn_TX_EN | RB_UEPn_BUF_MOD | Description: Arrange from low to high with R32_UEPn_DMA as the start address |
|---|---|---|---|
| 0 | 0 | x | Endpoint is disabled, and the UEPn_*_DMA buffer is not used. |
| 1 | 0 | 0 | The first address of receiver (OUT) buffer is R32_UEPn_RX_DMA |
| 1 | 0 | 1 | RB_UEP_R_TOG_MASK=0, use buffer R32_UEPn_RX_DMA<br>RB_UEP_R_TOG_MASK=1, use buffer R32_UEPn_TX_DMA |
| 0 | 1 | 0 | The first address of transmitter (IN) buffer is R32_UEPn_TX_DMA. |

| | | | |
|---|---|---|---|
| 0 | 1 | 1 | RB_UEP_T_TOG_MASK=0, use buffer R32_UEPn_TX_DMA<br>RB_UEP_T_TOG_MASK=1, use buffer R32_UEPn_RX_DMA |

*Note: The configuration options in the above table support n=1-7. Endpoint8 to endpoint15 configuration maps to endpoint1 to endpoint7 configuration.*

Start address of USB endpoint0 receive/transmit buffer (R32_UEP0_RT_DMA)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | UEP0_RT_DMA | RW | Start address of endpoint0 receive/transmit buffer, the lower 4 bits are fixed to 0 (16 bytes are aligned). | X |

*Note: DMA addresses RAMX area.*

Start address of USB endpoint n receive buffer (R32_UEPn_RX_DMA) (n=1-7(8-15))

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | UEPn_RX_DMA | RW | Start address of endpoint n receive buffer, the lower 4 bits are fixed to 0 (16 bytes are aligned). | X |

*Note: DMA addresses RAMX area.*

Start address of USB endpoint n transmit buffer (R32_UEPn_TX_DMA) (n=1-7(8-15))

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | UEPn_TX_DMA | RW | Start address of endpoint n transmit buffer, the lower 4 bits are fixed to 0 (16 bytes are aligned). | X |

*Note: DMA addresses RAMX area.*

Endpoint n reception maximum length packet register (R16_UEPn_MAX_LEN) (n=0-7(8-15))

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | UEPn_MAX_LEN | RW | Length of packet to be received by endpoint n. This register is used to prevent hardware DMA receive data from overflowing SRAM boundary. | X |

*Note: This maximum packet length determines the maximum length of data that can be received by the endpoint. The data beyond this length will be discarded and will not be written into the buffer.*

Endpoint n transmission length register (R16_UEPn_T_LEN) (n=0-7(8-15))

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | UEPn_T_LEN | RW | Set the number of bytes of data to be sent by USB endpoint n, the lower 11 bits are valid, and the higher 5 bits are fixed to 0 | X |

Endpoint n transmit control register (R8_UEPn_TX_CTRL) (n=0-7(8-15))

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:6] | Reserved | RO | Reserved. | 0 |
| 5 | RB_UEP_T_AUTOTOG | RW | Synchronization trigger bit auto toggle enable bit:<br>1: After the data is successfully received, the corresponding synchronization trigger bit is automatically toggled;<br>0: Not toggled automatically, and can be switched manually.<br>*Note: Only supported by endpoints other than endpoint0, and isochronous transfer is not supported.* | 0 |
| [4:3] | RB_UEP_T_TOG_MASK | RW | Synchronization trigger bit of the transmitter (processing IN transactions) of USB endpoint n<br>00: Transmit DATA0;<br>01: Transmit DATA1;<br>10: Transmit DATA2;<br>11: Transmit MDATA. | 0 |
| 2 | RB_UEP_TRES_NO | RW | 1: No expected response, used to implement real-time/synchronous transfer of non-endpoint0. Ignore MASK_UEP_T_RES at this time;<br>0: Expected response. | 0 |
| [1:0] | RB_UEP_TRES_MASK | RW | Response control by the transmitter of endpoint n to IN transactions:<br>00: Data is ready and ACK is expected;<br>10: Response NAK, or busy;<br>11: Response STALL, or error. | 0 |

Endpoint n receive control register (R8_UEPn_RX_CTRL) (n=0-7(8-15))

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:6] | Reserved | RO | Reserved. | 0 |
| 5 | RB_UEP_R_AUTOTOG | RW | Synchronization trigger bit auto toggle enable bit:<br>1: After the data is successfully received, the corresponding synchronization trigger bit is automatically toggled;<br>0: Not toggled automatically, and can be switched manually.<br>*Note: Only supported by endpoints other than endpoint0, and isochronous transfer is not supported.* | 0 |
| [4:3] | RB_UEP_R_TOG_MASK | RW | Expected synchronization trigger bit of the receiver (processing OUT transactions) of USB endpoint n:<br>00: Expected DATA0;<br>01: Expected DATA1;<br>10: Expected DATA2;<br>11: Expected MDATA.<br>*Note: It is invalid for real-time/synchronous transfer.* | 0 |

| | | | | |
|---|---|---|---|---|
| 2 | RB_UEP_RRES_NO | RW | 1: No expected response, used to implement isochronous transfer of non-endpoint 0. Ignore MASK_UEP_R_RES at this time;<br>0: Expected response. | 0 |
| [1:0] | RB_UEP_RRES_MASK | RW | Control on the response to OUT transactions by the receiver of USB endpoint n:<br>00: Response ACK; -<br>10: Response NAK or busy;<br>11: Response STALL or error;<br>01: Response to NYET.<br>*Note: It is invalid for isochronous transfer.* | 0 |

### 13.5.3 USB host controller register

USB host controller is equipped with 1 set of bidirectional host endpoints, including a transmission endpoint OUT and a reception endpoint IN. The maximum data packet length is 1024 bytes, and the endpoints support control transfer, interrupt transfer, bulk transfer, and isochronous transfer.

Each USB transaction initiated by host endpoint always automatically sets the RB_USB_IF_TRANSFER interrupt flag after processing. The application program can directly query or query and analyze the R8_USB_INT_FG register in the USB interrupt service program, and perform corresponding processing according to each interrupt flag. In addition, if RB_USB_IF_TRANSFER is valid, it is required to continue to analyze the USB interrupt status register (RB_USB_INT_ST), and perform the corresponding processing according to the response PID identification (RB_HOST_RES_MASK) of the current USB transfer transaction.

If the RB_UH_R_TOG_MASK bit of IN transaction of host reception endpoint is set in advance, you can judge whether the synchronization trigger bit of the data packet received matches the synchronization trigger bit of the endpoint through RB_USB_ST_TOGOK. If they are synchronized, the data is valid. If they are not synchronized, the data should be discarded. After the USB transmit/receive interrupt is processed each time, the synchronization trigger bit of corresponding host endpoint should be modified correctly to synchronize the data packet sent next time and detect whether the data packet received next time is synchronized. In addition, RB_UEP_R_AUTOTOG can be set to automatically toggle the corresponding synchronization trigger bit after successful reception or transmission.

R16_UH_EP_PID is used to set the endpoint number of the target device being operated and the token PID packet identification of the USB transfer transaction. The data corresponding to the SETUP token and OUT token is provided by the host transmission endpoint. The data to be sent is in the R32_UH_TX_DMA buffer, and the length of the data to be sent is set in R16_UH_TX_LEN. The data corresponding to the IN token is returned by the target device to the host reception endpoint, the received data is stored in the R32_UH_RX_DMA buffer, and the length of the received data is stored in R16_USB_RX_LEN. The maximum packet length that can be received by the host endpoint needs to be written to the R16_UH_MAX_LEN register in advance.

Table 13-3 USB host registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R8_UHOST_CTRL | 0x01 | USB host control register | 0x00 |
| R8_UH_EP_MOD | 0x11 | USB host endpoint mode control register | 0x00 |
| R32_UH_RX_DMA | 0x1C | USB host receive buffer start address | 0x000XXXXX |

| | | | |
|---|---|---|---|
| R32_UH_TX_DMA | 0x3C | USB host transmit buffer start address | 0x000XXXXX |
| R16_UH_MAX_LEN | 0x58 | USB host reception maximum length packet register | 0xXXXX |
| R16_UH_EP_PID | 0x78 | USB host token setting register | 00 |
| R8_UH_RX_CTRL | 0x7B | USB host reception endpoint control register | 0x00 |
| R16_UH_TX_LEN | 0x7C | USB host transmission length register | 0xXXXX |
| R8_UH_TX_CTRL | 0x7E | USB host transmission endpoint control register | 0x00 |
| R16_UH_SPLIT_DATA | 0x80 | USB host transmit SPLIT packet data | 0xXXXX |

USB host control register (R8_UHOST_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | RB_UH_AUTOSOF_EN | RW | Auto generate SOF packet enable.<br>1: The host automatically generates SOF packet;<br>0: No SOF packet generated.<br>*Note: This bit is automatically cleared by hardware when the device is disconnected.* | 0 |
| [6:3] | Reserved | RO | Reserved. | 0 |
| 2 | RB_UH_BUS_RESUME | RW | In host mode, bus suspended, the host wakes up the device. | 0 |
| 1 | RB_UH_BUS_SUSPEND | RW | USB host transmit suspend signal. | 0 |
| 0 | RB_UH_BUS_RESET | RW | USB host transmit bus reset signal. | 0 |

*Note: The reset time is determined by the high level duration of RB_UH_BUS_RESET. If the host wakes up the device, it is determined by the RB_UH_BUS_RESUME edge, so it is only required to pull RB_UH_BUS_RESUME high and then low for wake-up.*

USB host endpoint mode control register (R8_UH_EP_MOD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | Reserved | RO | Reserved. | 0 |
| 6 | RB_UH_TX_EN | RW | Host transmission endpoint transmit (SETUP/OUT) enable bit:<br>1: Endpoint transmit enabled;<br>0: Endpoint transmit disabled. | 0 |
| [5:4] | Reserved | RO | Reserved. | 0 |
| 3 | RB_UH_RX_EN | RW | Host reception endpoint receive (IN) enable bit:<br>1:Endpoint receive enabled;<br>0: Endpoint receive disabled. | 0 |
| [2:0] | Reserved | RO | Reserved. | 0 |

Start address of USB host receive buffer (R16_UH_RX_DMA)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | UH_RX_DMA | RW | Start address of host endpoint data receive buffer. The lower 4 bits are fixed to 0 (16 bytes are aligned). | X |

*Note: DMA addresses RAMX area.*

Start address of USB host transmit buffer (R32_UH_TX_DMA)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | UH_TX_DMA | RW | Start address of host endpoint data transmit buffer. The lower 4 bits are fixed to 0 (16 bytes are aligned). | X |

*Note: DMA addresses RAMX area.*

USB host reception maximum length packet register (R16_UH_MAX_LEN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | UH_MAX_LEN | RW | Maximum packet length of data received by the host endpoint. The lower 11 bits are valid, and the higher 5 bits are fixed to 0. | X |

*Note: This maximum packet size determines the maximum length of data that can be received by the endpoint. The data beyond this length will be discarded and DMA will not be sent to the custom area.*

USB host token setting register (R16_UH_EP_PID)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:8] | Reserved | RO | Reserved. | 0 |
| [7:4] | RB_UH_TOKEN_MASK | RW | Set the token PID packet identification of this USB transfer transaction. | 0 |
| [3:0] | RB_UH_EPNUM_MASK | RW | Set the endpoint number of the target device being operated this time. | 0 |

USB host reception endpoint control register (R8_UH_RX_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | Reserved | RO | Reserved. | 0 |
| 6 | RB_UH_RDATA_NO | RW | 1: Data packet not expected, used for high-speed HUB operation in host mode;<br>0: Data packet expected (IN). | 0 |
| 5 | RB_UH_R_AUTOTOG | RW | Synchronization trigger bit auto toggle enable.<br>1: After the data is successfully received, the corresponding expected synchronization trigger bit is automatically toggled;<br>0: Not toggled automatically, and can be switched manually. | 0 |
| [4:3] | RB_UH_R_TOG_MASK | RW | Synchronization trigger bit expected by the host receiver (processing IN transactions).<br>00: Expected DATA0;<br>01: Expected DATA1;<br>10: Expected DATA2;<br>11: Expected MDATA. | 0 |
| 2 | RB_UH_RRES_NO | RW | 1: No response, used to implement isochronous | 0 |

| | | | | |
|---|---|---|---|---|
| | | | transfer of non-endpoint 0. Ignore MASK_UEP_R_RES at this time;<br>0: Transmit response after data is received successfully. | |
| [1:0] | RB_UH_RRES_MASK | RW | Control on response to IN transactions by the receiver of host:<br>00: Response ACK; -<br>It is invalid for real-time/synchronous transfer. | 0 |

USB host transmission length register (R16_UH_TX_LEN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | UH_TX_LEN | RW | Set the number of bytes of data to be sent by USB host transmission endpoint, only the lower 11 bits are valid, and the higher 5 bits are fixed to 0. | X |

USB host transmission endpoint control register (R8_UH_TX_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| 7 | Reserved | RO | Reserved. | 0 |
| 6 | RB_UH_TDATA_NO | RW | 1: Data packet not transmitted (PING/SPLIT);<br>0: Data packet transmitted (OUT/SETUP). | 0 |
| 5 | RB_UH_T_AUTOTOG | RW | Synchronization trigger bit auto toggle enable bit:<br>1: After the data is successfully received, the corresponding synchronization trigger bit is automatically toggled;<br>0: Not toggled automatically, and can be switched manually. | 0 |
| [4:3] | RB_UH_T_TOG_MASK | RW | Synchronization trigger bit prepared by USB host transmitter (processing SETUP/OUT transactions)<br>00: Transmit DATA0;<br>01: Transmit DATA1;<br>10: Transmit DATA2;<br>11: Transmit MDATA. | 0 |
| 2 | RB_UH_TRES_NO | RW | 1: No response, used to implement isochronous transfer of non-endpoint0. Ignore MASK_UEP_T_RES at this time;<br>0: Expect a response after data is transmitted successfully. | 0 |
| [1:0] | RB_UH_TRES_MASK | RW | Response control bit of USB host transmitter to SETUP/OUT transaction<br>00: Expected response ACK;<br>10: Expected response NAK, or busy;<br>11: Expected response STALL, or error;<br>01: Expected response to NYET.<br>It is invalid for isochronous transfer. | 0 |

USB host transmit SPLIT packet data (R16_UH_SPLIT_DATA)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:0] | UH_SPLIT_DATA | RW | Data content of SPLIT packet transmitted by the host endpoint. The lower 12 bits are valid, and the higher 4 bits are fixed to 0 | X |

# Chapter 14 SD/EMMC controller

The system is equipped with 1 set of SDIO controller host interface, the transfer clock can reach 96MHz, 1/4/8- line communication mode is supported, and it can be connected to SD/TF card, EMMC card and other devices. The application code can flexibly set the parameters such as various commands for data transmission and reception, response packets, the mode and length of valid data packets and dual buffer length switching limits.

## 14.1 Main features

- Support SD Physical Layer Specification Versions 1.0 and 2.0, and support UHS-I SDR50 mode of SD Specification Version 3.0 (forward compatibility)
- Conform to EMMC Card Specification Versions 4.4 and 4.5.1, compliant with Specification Version 5.0, and compatible with HS200 mode
- Communication mode supports 1-line, 4-line and 8-line mode
- The highest communication clock is 96MHz
- Data packet length, command format and response status that can be flexibly set
- Provide the hardware to automatically stop the clock function at the interval of data blocks
- Support SD card, SDIO card, EMMC card and other devices that comply with SD interface protocol
- Support hardware AES and SM4 algorithm encryption/decryption for interface data transfer
- DMA dual buffer function

## 14.2 Functional description

### 14.2.1 Communication clock

- Frequency configuration

The controller interface provides 2 clock modes: low-speed mode and high-speed mode. Generally, the SD protocol stipulates that a communication clock of about 400KHz is used for the preliminary initialization of the device with SD interface to ensure better communication compatibility. After the parameter information of the external device is obtained, it can be switched to a higher clock frequency for communication according to the configuration supported internally.

Set bit9 in the R16_EMMC_CLK_DIV register to select the clock mode, and set the [4:0] bits to set the frequency division factor and get the final SDIO interface output clock. Bit8 controls EMMC peripheral to output a clock signal to MSDCK pin. It is necessary to ensure that the IO mode of MSDCK pin is configured as push-pull output.

- Phase configuration

When communicating with external SDIO interface devices, signal sampling errors may occur due to factors such as higher clock frequency, hardware wiring and device characteristics.

The system SDIO controller host interface is equipped with output clock phase toggle (offset by 180°) function. The output clock signal can be physically toggled by writing 1 to the bit10 in the R16_EMMC_CLK_DIV register, but the internal clock of controller remains unchanged. In this way, the communication timing can be adjusted.

### 14.2.2 Command transmission and response

The 48-bit data packet with a fixed command length sent by the SDIO host interface to the slave is serially transmitted on the MCMD line. Some commands do not require slave response, while some commands require slave response to the answer packets or data packets of different lengths.

- General command format: start bit (0) + transfer bit (1) + command index + parameter + CRC7 + end bit (1)

The start bit, transfer bit, CRC7 and end bit are automatically filled by hardware. The command index and parameter change according to different commands, and they need to be transmitted to the hardware controller through application codes. Configure the parameter domain in the SD command by writing into the R32_EMMC_ARGUMENT register. Write to the [5:0] bits in the R16_EMMC_CMD_SET register to complete the filling of command index, and configure the [11:8] bits in the register with the hardware for response packet detection expected by this command. The write operation to the R16_EMMC_CMD_SET register triggers the hardware to complete the command packet sequence transfer on the MCMD signal line.

- Short response format: start bit (0) + transfer bit (1) + command index + parameter + CRC7 + end bit (1)
- Long response format: start bit (0) + transfer bit (1) + reserved domain (111111b) + data + end bit (1)

The response packets responding to the slave are divided into 48-bit packet and 136-bit packet according to the length, and the valid data bits for the user are 32 bits and 128 bits respectively. After the host command is sent, you can query the R16_EMMC_INT_FG register. If the RB_EMMC_IF_CMDDONE flag bit is set to 1, the R32_EMMC_RESPONSE register can be read to obtain the valid data in the slave interface response packet. If the response packet is abnormal, the corresponding flag of [2:0] bits in the R16_EMMC_INT_FG register will be set.

### 14.2.3 Continuously read multiple data blocks

For the operation of reading data (transmitted through the data line) in the SD protocol command, the following configuration is required:

1. Initialization: Configure the R8_EMMC_CONTROL register, to set the sampling edge of commands and data, the width of the data line transmitted/received, and enable the DMA function. Configure the RB_EMMC_TOCNT_MASK register to set the command timeout time and data timeout time. Configure the R32_EMMC_DMA_BEG1 register to set the DMA start address for storing the data read. Configure the R32_EMMC_BLOCK_CFG register to set the length of a single data block and the total number of data blocks that need to be read during transmission. Configure the R32_EMMC_TRAN_MODE register, to set the DMA transfer direction as "SD to controller", and optionally set whether the clock needs to be automatically stopped at the block transfer interval, to give the application code enough time for processing, and then software starts transmission.
2. Send a command to the SDIO slave interface device. It requires the slave device to transfer data from the data line to the host interface, such as CMD17 and CMD18 in the SD protocol.
3. The controller interface reads data from the data line: when the slave interface device responds to the host command, it outputs data from the data line based on the host clock signal if the device is ready for data transfer, and the start signal is at low level. The signal is sampled at host controller terminal and transferred to the RAMX area configured by the user through the internal DMA. Every time the data of the size set by RB_EMMC_BKSIZE_MASK is transferred, a data block transmission is completed. Meanwhile, the total number of internal transmission data blocks is reduced by 1, until all data block transmissions are completed, the internal DMA will no longer perform data transfer. In this process, the

hardware will set the RB_EMMC_IF_BKGAP flag every time a data block is read. When all the data blocks have been read, the hardware will set the RB_EMMC_IF_TRANDONE flag. In case of data timeout or CRC error of received data when the controller is waiting for receiving data, RB_EMMC_IF_DATTMO and RB_EMMC_IF_TRANERR flags will be set respectively. The application code can configure the corresponding interrupt control bit in the R16_EMMC_INT_EN register. When the corresponding interrupt flag is set, the interrupt service of EMMC can be triggered.

4. Detect status: The R32_EMMC_STATUS register may be used for querying the level status of data line MD0 and command line MCMD in real time, and make judgments at the timing sequence specified in some protocols. The number of data blocks successfully transferred by DMA in the current command transmission is recorded in MASK_BLOCK_NUM domain.

### 14.2.4 Continuously write multiple data blocks

1. Initialize: Configure the R8_EMMC_CONTROL register, to set the sampling edge of commands and data, the width of the data line received/transmitted, and enable DMA. Configure the RB_EMMC_TOCNT_MASK register to set the command timeout time and data timeout time. Configure the R32_EMMC_BLOCK_CFG register, to set the length of a single data block and the total number of data blocks to be read during transmission. Configure the R32_EMMC_TRAN_MODE register to set the DMA transfer direction as "controller to SD".

2. Send a command to the SDIO slave interface device. It requires the host interface to transfer data from the data line to the slave interface, such as CMD24 and CMD25 in the SD protocol.

3. The controller interface writes data to the data line: when the slave interface device responds to the host command, it responds to the packet and release the data line status (not low level), and at this time the host can drive the data line to output a data block. The host interface is started to drive the data line by writing to the R32_EMMC_DMA_BEG1 register or performing write operation on the R32_EMMC_WRITE_CONT register. The difference is that the hardware loads the value of the R32_EMMC_DMA_BEG1 register into the internal at this time when operating the R32_EMMC_DMA_BEG1 register, and fetch data from this address and send it to controller for transmission. Hardware fetches data backwards and sends it to controller at the position where the current DMA moves internally when performing write operation on the R32_EMMC_WRITE_CONT register. Therefore, it is required to write to the R32_EMMC_DMA_BEG1 register to continue outputting data through data line for changing the DMA address (not continuous with the previous one) of data. If it is continuous with the previous address for writing data, the hardware automatically fetches the data address to move. You can directly write to the R32_EMMC_WRITE_CONT register to make the controller continue outputting data through data line. Every time the data of the size set by RB_EMMC_BKSIZE_MASK is output, a data block transmission is completed. Meanwhile, the total number of internal transmission data blocks is reduced by 1, until all data block transmissions are completed, the internal DMA will no longer perform data transmission. In this process, the hardware will set the RB_EMMC_IF_BKGAP flag every time a data block is read. When all the data blocks have been read, the hardware will set the RB_EMMC_IF_TRANDONE flag. In case of data timeout or CRC error of received data when the controller is waiting for receiving data, RB_EMMC_IF_DATTMO and RB_EMMC_IF_TRANERR flags will be set respectively. The application code can configure the corresponding interrupt control bit in R16_EMMC_INT_EN. When the corresponding interrupt flag is set, the interrupt service of EMMC can be triggered.

*Note: When the controller interface writes a data block to an external device, it needs to wait for the command to be sent and the response is completed before setting the DMA address, because the hardware is notified to drive the data line after setting the operation of DMA address. At this time, if the command has not been sent, the device will not accept the data.*

*Note: During the transmission of a some command execution (including reading data and writing data), the hardware will not be included into the successful counts in case a CRC error (RB_EMMC_IF_TRANERR) occurs, nor will it decrease the total count of transmission blocks, and the transmission of the next block will be continued. However, DMA will still put the data block with CRC error into the corresponding SRAM address area. It is required to pay attention to the processing at this moment for application code.*

### 14.2.5 DMA dual buffer

The system provides DMA dual buffer function. It has 2 DMA address registers (R32_EMMC_DMA_BEG1 and R32_EMMC_DMA_BEG2), and the data block length threshold register (RB_EMMC_DMATN_CNT) for DMA switching. When the transmission of data block length threshold starting from the R32_EMMC_DMA_BEG1 register address is completed, the hardware will switch to the R32_EMMC_DMA_BEG2 setting address area for data transmission, and switch to the R32_EMMC_DMA_BEG1 area again when the transmission threshold is reached. This cycle is repeated, until the total number of data blocks configured by RB_EMMC_BKNUM_MASK is completed for transmission.

Every time a non-zero data is written to RB_EMMC_BKNUM_MASK in the R32_EMMC_BLOCK_CFG register, the data transmission is started, and meanwhile the hardware also points the DMA to the R32_EMMC_DMA_BEG1 address area. The flag setting conditions during transmission are the same as those described above. The application code can choose to enable the corresponding interrupt bit to trigger the EMMC interrupt service.

## 14.3 Register description

SD/EMMC register base address: 0x4000A000

Table 14-1 EMMC registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R16_EMMC_CLK_DIV | 38h | Clock configuration register | 0x0213 |
| R32_EMMC_ARGUMENT | 00h | Command parameter register | 0x00000000 |
| R16_EMMC_CMD_SET | 04h | Command setting register | 0x0000 |
| R32_EMMC_RESPONSE0 | 08h | Response parameter register 0 | 0x00000000 |
| R32_EMMC_RESPONSE1 | 0Ch | Response parameter register 1 | 0x00000000 |
| R32_EMMC_RESPONSE2 | 10h | Response parameter register 2 | 0x00000000 |
| R32_EMMC_RESPONSE3 | 14h | Response parameter register 3 | 0x00000000 |
| R32_EMMC_WRITE_CONT | 14h | Continue write enable register | 0x00000000 |
| R8_EMMC_CONTROL | 18h | Control register | 0x15 |
| R8_EMMC_TIMEOUT | 1Ch | Timeout count register | 0x0C |
| R32_EMMC_STATUS | 20h | Status register | 0x00000000 |
| R16_EMMC_INT_FG | 24h | Interrupt flag register | 0x0000 |
| R16_EMMC_INT_EN | 28h | Interrupt enable register | 0x0000 |
| R32_EMMC_DMA_BEG1 | 2Ch | DMA start address 1 register | 0x0000XXXX |
| R32_EMMC_BLOCK_CFG | 30h | Transfer block configuration register | 0x00000000 |

| | | | |
|---|---|---|---|
| R32_EMMC_TRAN_MODE | 34h | Transfer mode register | 0x00000000 |
| R32_EMMC_DMA_BEG2 | 3Ch | DMA start address 2 register | 0x0000XXXX |

Clock configuration register (R16_EMMC_CLK_DIV)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:11] | Reserved | RO | Reserved. | 0 |
| 10 | RB_EMMC_PHASEINV | RW | SDCK clock output phase inverted. | 0 |
| 9 | RB_EMMC_CLKMode | RW | Clock frequency mode selection bit:<br>1: High-speed mode, 25M-100MHz;<br>0: Low speed mode, 400KHz. | 1 |
| 8 | RB_EMMC_CLKOE | RW | SD physical clock signal line output enable:<br>1: Enabled, output the communication clock;<br>0: Disabled. | 0 |
| [7:5] | Reserved | RO | Reserved. | 0 |
| [4:0] | RB_EMMC_DIV_MASK | RW | SD controller clock (SDCLK) division factor:<br>When bSDCLK_Mode=1,<br>SDCLK = 480M/MASK_CLK_PRE.<br>When bSDCLK_Mode =0,<br>SDCLK = 480M/MASK_CLK_PRE/64.<br>The minimum value is 2. Writing 1 is equivalent to disable the SDC mode clock. | 13h |

Command parameter register (R32_EMMC_ARGUMENT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | EMMC_ARGUMENT | RW | 32-bit command parameter register. | 0 |

Command setting register (R16_EMMC_CMD_SET)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:12] | Reserved | RO | Reserved. | 0 |
| 11 | RB_EMMC_CKIDX | RW | Command index for check response:<br>1: Required; 0: Not required. | 0 |
| 10 | RB_EMMC_CKCRC | RW | CRC for check response:<br>1: Required; 0: Not required. | 0 |
| [9:8] | RB_EMMC_RPTY_MASK | RW | Expected response type:<br>00b: No response;<br>01b: The response length is 136 bits;<br>10b: The response length is 48 bits;<br>11b: The response length is 48 bits, and it is an R1b type response. | 0 |
| [7:6] | Reserved | RO | Reserved. | 0 |
| [5:0] | RB_EMMC_CMDIDX_MASK | RW | Index number of the command being sent currently. | 0 |

Response parameter register (R32_EMMC_RESPONSE)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | R32_EMMC_RESPONSE0 | RO | Response parameter register 0 | 0 |
| [63:32] | R32_EMMC_RESPONSE1 | RO | Response parameter register 1 | 0 |
| [95:64] | R32_EMMC_RESPONSE2 | RO | Response parameter register 2 | 0 |
| [127:96] | R32_EMMC_RESPONSE3 | R0 | Response parameter register 3 | 0 |
| [127:96] | R32_EMMC_WRITE_CONT | WO | Multiplexing the EMMC_RESPONSE3 register, which is used to start the write operation in the multi-block write process. | 0 |

*Note: When the response length is 136 bits, the effective data length is 128 bits. When the response length is 48 bits, the effective data length is 32 bits. The EMMC_RESPONSE3 register is used to store the effective data parameters for response.*

The EMMC_RESPONSE3 register has alternate function. In the process of continuously writing multiple blocks of data to the card using the CMD25 command and at the end of block transmission, if it is not needed to change DMA address, write operation on this register will start the current address to continue the operation of writing data. To change the DMA address, write to the DMA address register to start the write operation, with no need to start by writing to register.

Control register (R8_EMMC_CONTROL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:6] | Reserved | RO | Reserved. | 0 |
| 5 | RB_EMMC_NEGSMP | RW | Cmd and Data signal line sampling mode selection bits:<br>1: Sample on the falling edge;<br>1: Sample on the rising edge. | 0 |
| 4 | RB_EMMC_RST_LGC | RW | Internal logic receive/transmit reset.<br>1: Reset;<br>0: Work normally. | 1 |
| 3 | RB_EMMC_DMAEN | RW | Controller DMA enable:<br>1: DMA enabled;<br>0: DMA disabled. | 0 |
| 2 | RB_EMMC_ALL_CLR | RW | Internal FIFO and interrupt flag reset.<br>1: Reset;<br>0: Work normally. | 1 |
| [1:0] | RB_EMMC_LW_MASK | RW | Effective data width for receiving/transmitting data:<br>00: The transceiver only uses dat[0], single data line;<br>01: The transceiver uses dat[3:0], 4 data lines;<br>10: The transceiver uses dat[7:0], 8 data lines. | 01b |

Timeout control register (R8_EMMC_TIMEOUT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:4] | Reserved | RO | Reserved. | 0 |
| [3:0] | RB_EMMC_TOCNT_MASK | RW | Response/data timeout configuration.<br>Non-zero: Set the timeout time, and valid values | Ch |

| | | | | |
|---|---|---|---|---|
| | | | range from 0 to 12;<br>Calculation:<br>Module clock cycle * 4194304 * MASK_TOCNT.<br>For example: When SDCLK cycle is 10ns, write 12, and the timeout interval is 10ns * (4194304) * (12) = 503ms.<br>0: Internal timeout mechanism disabled. | |

*Note 1: The above data timeout includes the following 4 situations:*

*1) DAT[0] busy timeout after R1b response;*

*2) When writing a data block, DAT[0] busy timeout after CRC status;*

*3) When writing a data block, waiting for the CRC status timeout;*

*4) When reading a data block, waiting for the start bit timeout.*

*Note 2: The response to the command also supports timeout mechanism. If the response times out, it is given by the R16_EMMC_INT_FG interrupt in the interrupt register. The command timeout uses the maximum timeout value given by the protocol: 64Tsdclk.*

Status indication register (R32_EMMC_STATUS)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [32:18] | Reserved | RO | Reserved. | 0 |
| 17 | RB_EMMC_DAT0STA | RO | Current level status of DAT0 data line.<br>1: High level; 0: Low level. | 0 |
| 16 | RB_EMMC_CMDSTA | RO | Current level status of CMD signal line<br>1: High level; 0: Low level. | 0 |
| [15:0] | MASK_BLOCK_NUM | RO | The number of blocks that have been successfully transmitted in the current multi-block transmission operation. | 0 |

Interrupt flag register (R16_EMMC_INT_FG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:10] | Reserved | RO | Reserved. | 0 |
| 9 | RB_EMMC_IF_SDIOINT | RW1 | SDIO card interrupt flag. Cleared by writing 1.<br>1: SDIO card generates card interrupt;<br>0: No event. | 0 |
| 8 | RB_EMMC_IF_FIFO_OV | RW1 | FIFO overflow flag. Cleared by writing 1.<br>1: FIFO overflow trigger;<br>0: No event. | 0 |
| 7 | RB_EMMC_IF_BKGAP | RW1 | Single block transfer completion flag. Cleared by writing 1.<br>1: Single block reception/transmission completion triggered;<br>0: No event. | 0 |
| 6 | RB_EMMC_IF_TRANDONE | RW1 | All blocks transfer completion flag. Cleared by writing 1.<br>1: All blocks reception/transmission completion triggered<br>0: No event. | 0 |

| | | | | |
|---|---|---|---|---|
| 5 | RB_EMMC_IF_TRANERR | RW1 | Data transfer CRC error flag. Cleared by writing 1.<br>1: CRC error trigger;<br>0: No event. | 0 |
| 4 | RB_EMMC_IF_DATTMO | RW1 | Data timeout flag. Cleared by writing 1.<br>1: Data timeout triggered;<br>0: No event. | 0 |
| 3 | RB_EMMC_IF_CMDDONE | RW1 | Command completion flag. Cleared by writing 1.<br>1: Send the command, and receive the completion response;<br>0: No event. | 0 |
| 2 | RB_EMMC_IF_REIDX_ER | RW1 | Response index number check error flag. Cleared by writing 1.<br>1: Response index number check error triggered;<br>0: No event. | 0 |
| 1 | RB_EMMC_IF_RECRC_WR | RW1 | Response CRC check error flag. Cleared by writing 1.<br>1: Response CRC check error triggered;<br>0: No event. | 0 |
| 0 | RB_EMMC_IF_RE_TMOUT | RW1 | Receive response timeout flag. Cleared by writing 1.<br>1: Response timeout triggered;<br>0: No event. | 0 |

Interrupt enable register (R16_EMMC_INT_EN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:10] | Reserved | RO | Reserved. | 0 |
| 9 | RB_EMMC_IE_SDIOINT | RW | SDIO card interrupt enable.<br>1: SDIO card interrupt enabled;<br>0: SDIO card interrupt disabled. | 0 |
| 8 | RB_EMMC_IE_FIFO_OV | RW | FIFO overflow interrupt enable.<br>1: FIFO overflow interrupt enabled;<br>0: FIFO overflow interrupt disabled. | 0 |
| 7 | RB_EMMC_IE_BKGAP | RW | Single block transfer completed interrupt enable.<br>1: Single block transfer completed interrupt enabled;<br>0: Single block transfer completed interrupt disabled. | 0 |
| 6 | RB_EMMC_IE_TRANDONE | RW | All block transfer completion interrupt enable.<br>1: All block transfer completion interrupt enabled;<br>0: All block transfer completion interrupt disabled. | 0 |
| 5 | RB_EMMC_IE_TRANERR | RW | Block transfer CRC error interrupt:<br>1: Block transfer CRC error interrupt enabled;<br>0: Block transfer CRC error interrupt disabled. | 0 |
| 4 | RB_EMMC_IE_DATTMO | RW | Data timeout interrupt:<br>1: Data timeout interrupt enabled;<br>0: Data timeout interrupt disabled. | 0 |
| 3 | RB_EMMC_IE_CMDDONE | RW | Command completion interrupt:<br>1: Command completion interrupt enabled;<br>0: Command completion interrupt disabled. | 0 |

| | | | | |
|---|---|---|---|---|
| 2 | RB_EMMC_IE_REIDX_ER | RW | Response index check error interrupt:<br>1: Response index check error interrupt enabled;<br>0: Response index check error interrupt disabled. | 0 |
| 1 | RB_EMMC_IE_RECRC_WR | RW | Response CRC check error interrupt:<br>1: Response CRC check error interrupt enabled;<br>0: Response CRC check error interrupt disabled. | 0 |
| 0 | RB_EMMC_IE_RE_TMOUT | RW | Command response timeout interrupt:<br>1: Command response timeout interrupt enabled;<br>0: Command response timeout interrupt disabled. | 0 |

Data block DMA start address register (R32_EMMC_DMA_BEG1)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | RB_EMMC_DMAAD1_MASK | RW | Read/write the start address of data buffer. The lower 4 bits are fixed to 0 (16 bytes are aligned). | 0 |

*Note: When reading data from SD, this register stores the start address of the read data in SRAM. When writing data to the SD card, this register stores the start address of the data to be written in SRAM. Addressing is in RAMX area.*

If continuous multi-block read/write SD operations are performed, the user can write to the DMA_BEG1 register to change the DMA address as needed after the single block transmission is completed (RB_EMMC_IF_BKGAP). Do not change the DMA address during transfer, otherwise, data count errors may be caused. When performing continuous multi-block write operations, it is required to start the continued write operation by writing to the R32_EMMC_WRITE_CONT or DMA_BEG1 register after the single block transmission is completed. It is not required when performing multi-block read operations.

Transfer block configuration register (R32_EMMC_BLOCK_CFG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:28] | Reserved | RO | Reserved. | 0 |
| [27:16] | RB_EMMC_BKSIZE_MASK | RW | Single block transfer size (1-2048 bytes). | 0 |
| [15:0] | RB_EMMC_BKNUM_MASK | RW | Count of blocks (1~65535 blocks) to be transmitted by DMA this time. Automatically cleared internally. If the number of blocks is not zero, the receiving or transmitting is enabled. | 0 |

Transfer mode register (R32_EMMC_TRAN_MODE)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| 16 | RB_EMMC_DULEDMA_EN | RW | DMA dual buffer function enable. | 0 |
| 15 | Reserved | RO | Reserved. | 0 |
| [14:8] | RB_EMMC_DMATN_CNT | RW | In dual buffer mode, set the block count value for buffer switching. | 10h |
| [7:6] | RB_EMMC_FIFO_RDY | RW | FIFO ready selection signal when writing to EMMC to adjust the relationship between internal FIFO and DMA. | 01b |

| | | | | |
|---|---|---|---|---|
| 5 | Reserved | RW | Must write 0. | 0 |
| 4 | RB_EMMC_AUTOGAPSTOP | RW | Hardware auto data block stop clock enable. | 0 |
| 3 | Reserved | RW | Must write 0. | 0 |
| 2 | RB_EMMC_MODE_BOOT | RW | Set the eMMC card transfer mode:<br>1: Boot mode;<br>0: Normal mode.<br>*Note: Only for eMMC card.* | 0 |
| 1 | RB_EMMC_GAP_STOP | RW | Data block completion clock stop mode. When block data is read and this bit is set to 1, the hardware automatically disable the clock output after sampling of a data block count and a new block transmission is started. Cleared by software, and set to 1 for counting again after the delay of single SCLK.<br>If RB_EMMC_AUTOGAPSTOP is enabled, this bit is set to 1 by hardware and only needs to be cleared by software. | 0 |
| 0 | RB_EMMC_DMA_DIR | RW | Direction of DMA transfer:<br>1: Controller to SD;<br>0: SD to controller. | 0 |

Data block DMA start address register (R32_EMMC_DMA_BEG2)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:17] | Reserved | RO | Reserved. | 0 |
| [16:0] | RB_EMMC_DMAAD2_MASK | RW | Read/write the start address of data buffer. The lower 4 bits are fixed to 0 (16 bytes are aligned). | 0 |

*Note: When reading data from SD, this register stores the start address of the read data in SRAM. When writing data to the SD card, this register stores the start address of the data to be written in SRAM. Addressing is in RAMX area.*

# Chapter 15 Encryption module (ECDC)

The system is equipped with a built-in block cipher algorithm module, which supports 2 types of block cipher algorithms (AES and SM4), and Electronic Codebook (ECB) and Counter (CTR) modes. The module can complete single encryption/decryption process with 128-bit data size as the basic unit, and it is provided with hardware automatic encryption/decryption modes for data transfer between some peripheral interfaces and SRAM and hardware encryption/decryption modes for its own SRAM data.

## 15.1 Main features

- ECB mode and CTR mode with SM4 algorithm 128-bit key
- ECB mode and CTR mode with AES algorithm 128/192/256 bit key
- Support data encryption and decryption from memory to memory, memory to HSPI, and memory to EMMC
- Support to directly encrypt single 128-bit data by software in the way of writing to SFR
- Support to encrypt and decrypt data blocks of length specified by software in the way of DMA

## 15.2 AES/SM4 algorithm

The AES (Advanced Encryption Standard) algorithm is a block cipher that uses a symmetric block cipher system, which is one of the most popular algorithms in symmetric key encryption. The SM4 block cipher algorithm is generally a special block cipher for wireless local area networks and trusted computers, and it can also be used for data encryption protection in other environments.

In the process of data encryption and decryption, the key needs to be loaded. For the AES algorithm, the key length is set to 128/192/256 bits. When using it, the user key needs to be extended to 11×128/13×128/15×128-bit extended keys. The SM4 algorithm expands the 128-bit user key into a 32×32-bit extended key. These extended keys are stored in internal registers for use during encryption and decryption.

## 15.3 ECB and CTR mode

AES/SM4 supports 2 modes, including electronic codebook (ECB) mode and counter (CTR) mode. The security performance of CTR mode is higher than that of ECB mode. See Figure 16-1 and Figure 16-2 for the working block diagram.

Figure 15-1 Encryption and decryption in ECB mode

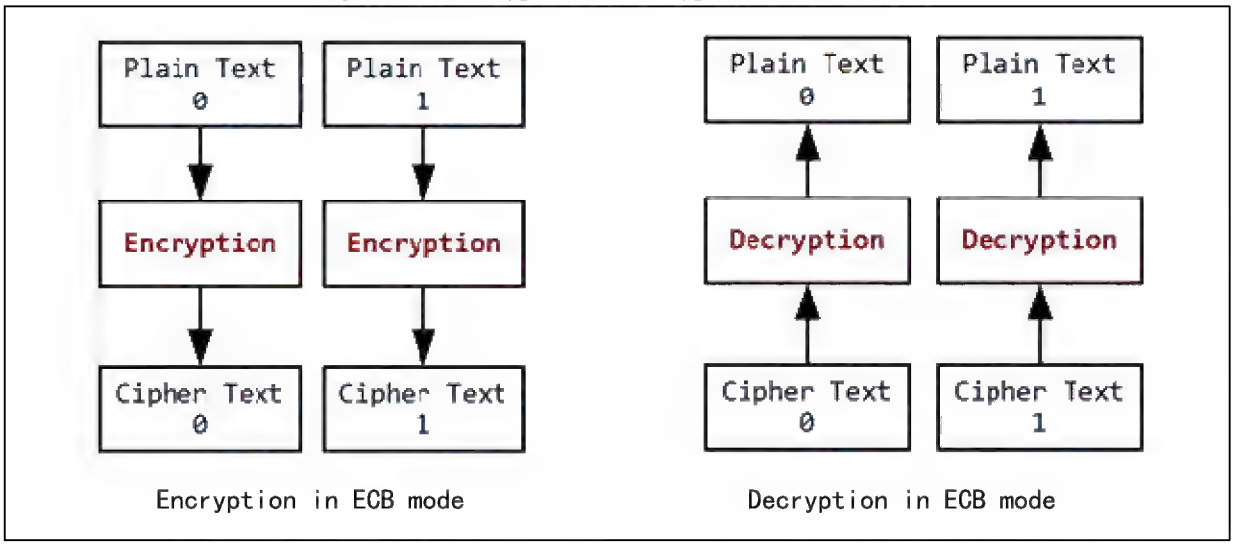
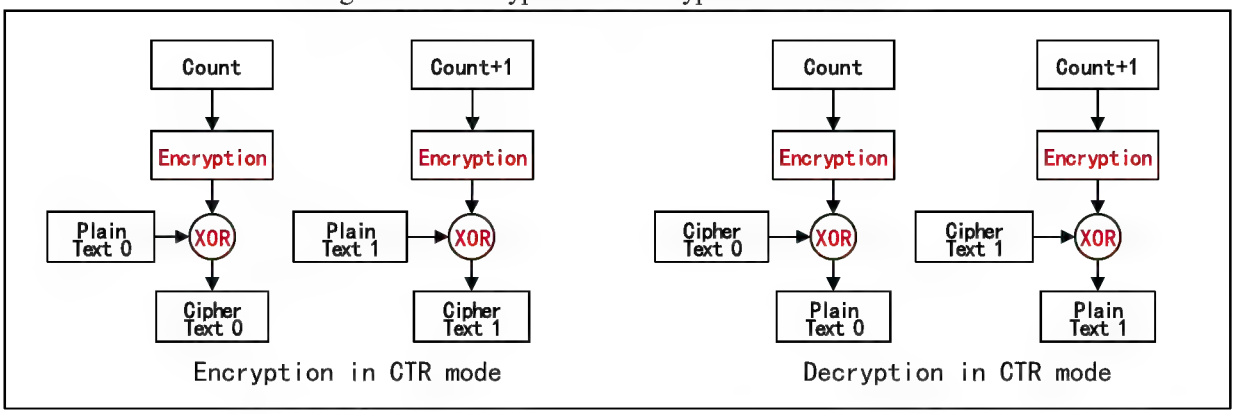



Figure 15-2 Encryption and decryption in CTR mode




In ECB mode, there is a one-to-one correspondence between plain text and cipher text, and the encrypted plain text is directly used as cipher text. While in CTR mode, a 128-bit count value needs to be loaded in advance to encrypt the count value. And XOR logic is used between the encrypted count value and the plain text, then used as the cipher text. It is worth noting that in CTR decryption mode, the count value is only encrypted, but not decrypted.

## 15.4 Application

### 15.4.1 Encryption and decryption module initialization

- Clock configuration: Set the [5:4] bits in the R16_ECEC_CTRL register to configure the clock for module operation. If this module is not used, these bits can be set to 0, to disable the module and reduce power consumption.
- Data big/little endian: Set bit13 in the R16_ECEC_CTRL register, to select the big/little endian mode for encrypted/decrypted data storage. It is stored at the packet size end of128bits (16 bytes).
- Key expansion: Extended key is not used in the encryption and decryption process but not the user key. After setting the user key length and content, it is required to notify the hardware to perform the key expansion action and the key will be saved to the internal register of the module after expansion. After that, data encryption and decryption work can be conducted properly. Set the [11:8] bits in the

R16_ECEC_CTRL register, to select the encryption and decryption algorithm, block cipher mode and key length. Then configure the user key register group R32_ECDC_KEY, and fill in the user key content. Set bit0 in the R16_ECEC_CTRL register to 1 and then clear it. After that, hardware starts to perform the key expansion function. It is required to query the RB_ECDC_IF_EKDONE bit in the R8_ECDC_INT_FG register through application software. If it is set to 1, the key expansion is completed, and it is required to clear the RB_ECDC_IF_EKDONE bit status. If the RB_ECDC_IE_EKDONE function of R8_ECDC_INT_EN register is enabled, the interrupt service is triggered when the RB_ECDC_IF_EKDONE flag is set.

- Initial value of counter: If CTR mode is used for encryption and decryption, an initial value of counter is also required. Set the CTR mode count value register group R32_ECDC_IV and fill in the 128-bit count value. Each time the data encryption and decryption (128bit data) module is successfully completed, the internal counter value of module will increase by 1. This counter value is not required for ECB mode.

### 15.4.2 Several encryption and decryption configurations

The encryption and decryption data flow of module includes the following methods: memory to memory, memory to HSPI, memory to EMMC and single 128bit data encryption and decryption. Please refer to the configurations in Table 15-1 for details.

Table 15-1 Configurations of different data stream modes

| Encryption and decryption data stream | RB_ECDC_ WRSRAM_EN | RB_ECDC_ RDPERI_EN | RB_ECDC_ WRPERI_EN | RB_ECDC_ MODE_SEL | Incidentals |
|---|---|---|---|---|---|
| RAMX to peripheral encryption | 0 | 0 | 1 | 0 | Once the encryption and decryption function is enabled, it will be converted during the data transfer process. The address of RAMX is the DMA address configured in the peripheral. |
| RAMX to peripheral decryption | 0 | 0 | 1 | 1 | |
| Peripheral to RAMX encryption | 0 | 1 | 0 | 0 | |
| Peripheral to RAMX decryption | 0 | 1 | 0 | 1 | |
| RAMX data encryption | 1 | 1 | 0 | 0 | It is required to configure the R32_ECDC_SRAM_ADDR register and the R32_ECDC_SRAM_LEN register |
| | | 0 | 1 | | |
| RAMX data decryption | 1 | 1 | 0 | 1 | |
| | | 0 | 1 | | |
| Single encryption of 128-bit data | - | 1 | 0 | 0 | Write to the R32_ECDC_SGSD register to fill in the original data (plain text or cipher text) Read the R32_ECDC_SGRT register, to read the result data after encryption and decryption. |
| | | 0 | 1 | | |
| Single decryption of 128-bit data | - | 1 | 0 | 1 | |
| | | 0 | 1 | | |

*Note: Peripherals include HSPI and EMMC modules.*

See the modes described in Table 15-1. The data conversion between memory and peripherals is automatically interspersed by hardware during data transfer. The transfer start and end flags of peripheral function interface also represent the start and completion of encryption and decryption conversion.

The data conversion from memory to memory is enabled by writing non-0 data into the R32_ECDC_SRAM_LEN length register. Query the RB_ECDC_IF_WRSRAM bit in the R8_ECDC_INT_FG register. If it is set to 1, it means that data conversion from memory to memory is completed, and it is required to clear the RB_ECDC_IF_WRSRAM bit status. If the RB_ECDC_IE_WRSRAM in the R8_ECDC_INT_EN register is enabled, the interrupt service is triggered when the RB_ECDC_IF_WRSRAM flag is set.

The single encryption and decryption mode must be configured as big-endian (RB_ECDC_DAT_MOD=1) for execution. The single encryption/decryption conversion of 128-bit data starts to write the operation initiation to the highest 32 bits in the R32_ECDC_SGSD register. Query the RB_ECDC_IF_SINGLE bit in the R8_ECDC_INT_FG register. If it is set to 1, the single 128-bit data conversion is completed, and the RB_ECDC_IF_SINGLE bit status is cleared. If the RB_ECDC_IE_SINGLE in the R8_ECDC_INT_EN register is enabled, the interrupt service is triggered when the RB_ECDC_IF_SINGLE flag is set.

## 15.5 Register description

ECDC register physical base address: 0x40007000

Table 15-2 ECDC registers

| Name | Offset address | Description | Reset value |
|---|---|---|---|
| R16_ECEC_CTRL | 0x00 | ECDC control register | 0x0020 |
| R8_ECDC_INT_EN | 0x02 | ECDC interrupt enable register | 0x00 |
| R8_ECDC_INT_FG | 0x06 | ECDC interrupt flag register | 0x00 |
| R32_ECDC_KEY_255T224 | 0x08 | User key register 7 | 0xXXXXXXXX |
| R32_ECDC_KEY_223T192 | 0x0C | User key register 6 | 0xXXXXXXXX |
| R32_ECDC_KEY_191T160 | 0x10 | User key register 5 | 0xXXXXXXXX |
| R32_ECDC_KEY_159T128 | 0x14 | User key register 4 | 0xXXXXXXXX |
| R32_ECDC_KEY_127T96 | 0x18 | User key register 3 | 0xXXXXXXXX |
| R32_ECDC_KEY_95T64 | 0x1C | User key register 2 | 0xXXXXXXXX |
| R32_ECDC_KEY_63T32 | 0x20 | User key register 1 | 0xXXXXXXXX |
| R32_ECDC_KEY_31T0 | 0x24 | User key register 0 | 0xXXXXXXXX |
| R32_ECDC_IV_127T96 | 0x28 | CTR mode count value register 3 | 0xXXXXXXXX |
| R32_ECDC_IV_95T64 | 0x2C | CTR mode count value register 2 | 0xXXXXXXXX |
| R32_ECDC_IV_63T32 | 0x30 | CTR mode count value register 1 | 0xXXXXXXXX |
| R32_ECDC_IV_31T0 | 0x34 | CTR mode count value register 0 | 0xXXXXXXXX |
| R32_ECDC_SGSD_127T96 | 0x40 | Original data 3 of single encryption and decryption | 0x00000000 |
| R32_ECDC_SGSD_95T64 | 0x44 | Original data 2 of single encryption and decryption | 0x00000000 |
| R32_ECDC_SGSD_63T32 | 0x48 | Original data 1 of single encryption and decryption | 0x00000000 |
| R32_ECDC_SGSD_31T0 | 0x4C | Original data 0 of single encryption and decryption | 0x00000000 |
| R32_ECDC_SGRT_127T96 | 0x50 | Single encryption and decryption result 3 | 0x00000000 |

| | | | |
|---|---|---|---|
| R32_ECDC_SGRT_95T64 | 0x54 | Single encryption and decryption result 2 | 0x00000000 |
| R32_ECDC_SGRT_63T32 | 0x58 | Single encryption and decryption result 1 | 0x00000000 |
| R32_ECDC_SGRT_31T0 | 0x5C | Single encryption and decryption result 0 | 0x00000000 |
| R32_ECDC_SRAM_ADDR | 0x60 | Start address of encryption and decryption SRAM area | 0x00000000 |
| R32_ECDC_SRAM_LEN | 0x64 | Encryption and decryption SRAM length (/128bit) | 0x00000000 |

ECDC control register (R16_ECEC_CTRL)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [15:14] | Reserved | RO | Reserved. | 0 |
| 13 | RB_ECDC_DAT_MOD | WO | Encryption and decryption data big/little endian mode selection.<br>1: Big-endian mode; 0: Little-endian mode.<br>*Note: It must be configured as big-endian mode for single 128-bit encryption and decryption.* | 0 |
| 12 | Reserved | RO | Reserved. | 0 |
| [11:10] | RB_ECDC_KLEN_MASK | RW | Key length setting.<br>00: 128-bit; 01: 192-bit;<br>10: 256-bit; 11: Reserved. | 0 |
| 9 | RB_ECDC_CIPHER_MOD | RW | Block cipher mode selection.<br>1: CTR mode; 0: ECB mode. | 0 |
| 8 | RB_ECDC_ALGRM_MOD | RW | Encryption and decryption algorithm mode selection.<br>1: AES; 0: SM4. | 0 |
| 7 | RB_ECDC_WRSRAM_EN | RW | Enable SRAM data encryption and decryption. It is required to use bit2/bit1 together.<br>0: Enabled; 1: Disabled. | 0 |
| [6:4] | RB_ECDC_CLKDIV_MASK | RW | Encryption and decryption clock frequency division factor.<br>Calculation: EDclk=480M/ED_CLK_PRE.<br>The minimum value is 2. Writing 1 is equivalent to disabling ECDC module operation clock. | 10b |
| 3 | RB_ECDC_MODE_SEL | RW | Encryption and decryption mode selection.<br>1: Decryption mode; 0: Encryption mode. | 0 |
| 2 | RB_ECDC_WRPERI_EN | RW | Enable data from SRAM to peripherals for encryption and decryption control.<br>1: Encryption and decryption enabled;<br>0: Not work. | 0 |
| 1 | RB_ECDC_RDPERI_EN | RW | Enable data from peripheral to SRAM for encryption and decryption control.<br>1: Encryption and decryption enabled;<br>0: Not work. | 0 |
| 0 | RB_ECDC_KEYEX_EN | RW | Key extension function enable control bit. Enabled by high level pulse. After the key length | 0 |

| | | | | |
|---|---|---|---|---|
| | | | is configured, it is required to set this bit to 1, and wait until the key expansion is completed, then perform the encryption and decryption process.<br>*Note: The application code driver needs to set it to high and then low.* | |

ECDC interrupt enable register (R8_ECDC_INT_EN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:3] | Reserved | RO | Reserved. | 0 |
| 2 | RB_ECDC_IE_WRSRAM | WO | Encryption and decryption from memory to memory completion interrupt enable.<br>1: Encryption and decryption from memory to memory completed interrupt enabled;<br>0: Encryption and decryption from memory to memory completed interrupt disabled. | 0 |
| 1 | RB_ECDC_IE_SINGLE | RW | Single encryption and decryption completion interrupt enable. The conversion is completed for 128-bit registers.<br>1: Single encryption and decryption completion interrupt enabled;<br>0: Single encryption and decryption completion interrupt disabled. | 0 |
| 0 | RB_ECDC_IE_EKDONE | RW | Key expansion completion interrupt enable.<br>1: Key expansion completion interrupt enabled;<br>0: Key expansion completion interrupt disabled. | 0 |

ECDC interrupt flag register (R8_ECDC_INT_FG)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [7:3] | Reserved | RO | Reserved. | 0 |
| 2 | RB_ECDC_IF_WRSRAM | RW1 | Memory-to-memory encryption and decryption completion interrupt flag. Cleared by writing 1.<br>1: Encryption and decryption completion event;<br>0: No event. | 0 |
| 1 | RB_ECDC_IF_SINGLE | RW1 | Single encryption and decryption completion interrupt flag. Cleared by writing 1.<br>1: Single encryption and decryption completion event;<br>0: No event. | 0 |
| 0 | RB_ECDC_IF_EKDONE | RW1 | Key expansion completion interrupt flag. Cleared by writing 1.<br>1: Key expansion completion event;<br>0: No event. | 0 |

User key register group (R32_ECDC_KEY)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | R32_ECDC_KEY_255T224 | RW | User key bit224 to bit256. | X |
| [31:0] | R32_ECDC_KEY_223T192 | RW | User key bit192 to bit223. | X |
| [31:0] | R32_ECDC_KEY_191T160 | RW | User key bit160 to bit191. | X |
| [31:0] | R32_ECDC_KEY_159T128 | RW | User key bit128 to bit159. | X |
| [31:0] | R32_ECDC_KEY_127T96 | RW | User key bit96 to bit127. | X |
| [31:0] | R32_ECDC_KEY_95T64 | RW | User key bit64 to bit95. | X |
| [31:0] | R32_ECDC_KEY_63T32 | RW | User key bit32 to bit63. | X |
| [31:0] | R32_ECDC_KEY_31T0 | RW | User key bit0 to bit31. | X |

CTR mode count value register group (R32_ECDC_IV)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | R32_ECDC_IV_127T96 | RW | Count value bit96 to bit127. | X |
| [31:0] | R32_ECDC_IV_95T64 | RW | Count value bit64 to bit95. | X |
| [31:0] | R32_ECDC_IV_63T32 | RW | Count value bit32 to bit63. | X |
| [31:0] | R32_ECDC_IV_31T0 | RW | Count value bit0 to bit31. | X |

Single encryption and decryption of original data register group (R32_ECDC_SGSD)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | R32_ECDC_SGSD_127T96 | RW | Original data bit96 to bit127. | 0 |
| [31:0] | R32_ECDC_SGSD_95T64 | RW | Original data bit64 to bit95. | 0 |
| [31:0] | R32_ECDC_SGSD_63T32 | RW | Original data bit32 to bit63. | 0 |
| [31:0] | R32_ECDC_SGSD_31T0 | RW | Original data bit0 to bit31. | 0 |

*Note: Internal encryption and decryption are performed at a fixed size of 128 bits. When the write operation on the R32_ECDC_SGSD_31T0 register is completed, the hardware automatically starts a single encryption/decryption conversion. RB_ECDC_IF_SINGLE indicates whether the conversion is completed.*

Single encryption and decryption result register group (R32_ECDC_SGRT)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:0] | R32_ECDC_SGRT_127T96 | RW | Data bit96 to bit127. | 0 |
| [31:0] | R32_ECDC_SGRT_95T64 | RW | Data bit64 to bit95. | 0 |
| [31:0] | R32_ECDC_SGRT_63T32 | RW | Data bit32 to bit63. | 0 |
| [31:0] | R32_ECDC_SGRT_31T0 | RW | Data bit0 to bit31. | 0 |

*Note: The encryption and decryption result data is stored in fixed big-endian mode, with 128bits structure.*

Start address of encryption and decryption SRAM area (R32_ECDC_SRAM_ADDR)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [32:17] | Reserved | RO | Reserved. | 0 |

| | | | | |
|---|---|---|---|---|
| [16:0] | ECDC_SRAM_ADDR | RW | Start address of encryption and decryption SRAM data, the lower 4 bits are fixed to 0 (16 bytes are aligned). | 0 |

*Note: This DMA addressing is in the RAMX area.*

Encryption and decryption SRAM length (R32_ECDC_SRAM_LEN)

| Bit | Name | Access | Description | Reset value |
|---|---|---|---|---|
| [31:13] | Reserved | RO | Reserved. | 0 |
| [12:0] | ECDC_SRAM_LEN | RW | Encryption and decryption SRAM data length, in the unit of 128bit. When a non-zero value is written, the encryption and decryption process of SRAM data is enabled. | 0 |

# Chapter 16 Serializer and Deserializer (SerDes)

The system has a built-in SerDes module that supports signal isolation and long-distance transmission. The SerDes module supports 1.2Gbps high-speed differential signals (GXM/GXP pins). Long-distance data transmission can be implemented through a differential peer transmission medium in the optical fiber module or network cable.

SerDes module peripheral base address: 0x4000B000

## Main features

- Programmable data reception/transmission rate, up to 1.2Gbps
- Built-in 8b/10b codec and CRC check. Support sequence number matching
- Built-in FIFO, which supports dual buffer mode for transmission and reception
- Support DMA function; access address supports byte alignment
- Provide several transmission interrupt flags and states, and send feedback information to the application layer in a timely manner
- Differential transceiver, which can directly drive optical fiber module
- The measured transfer distance is about 70 meters using a pair of category 5e differential network cables at a rate of 600Mbps
- The measured transfer distance is about 90 meters using a pair of category 6 differential network cables at a rate of 600Mbps
- The measured transfer distance is about 25 meters using a pair of category 6 differential network cables at a rate of 1.2Gbps
- The measured transfer distance is about 100 meters using a pair of category 6 differential network cables at a rate of 360Mbps

Please refer to and call the provided subroutine library for specific applications.

# Chapter 17 USB3.0 controller and transceiver (USBSS)

The USB3.0 controller includes a USB host controller and a USB device controller. With the built-in physical transceiver PHY, it can implement the functions of USB3.0 interface products. It supports 5Gbps USB SuperSpeed signal, and hardware interface includes 2 pairs of super high speed differential signal lines (SSRXA/SSRXB and SSTXA/SSTXB, A/B can be connected to +/- or -/+).

This controller provides a link layer register access interface for application code to manage device connection/disconnection, bus status and power mode. It provides a HOST access interface and a DEVICE access interface, which are used to implement various data transmission and upper layer protocols in the USB3.0 protocol specification.

USBSS module peripheral base address: 0x40008000

## Main features

- Supports USB3.0 protocol interface specification
- Supports USB Host function and USB Device function
- Supports OTG function
- Supports drive USB3.0 HUB
- Both the host and the device support control transfer, bulk transfer, interrupt transfer, isochronous transfer
- Directly access the data of each endpoint buffer through DMA
- Power management, support U1/U2/U3 low power state
- All endpoints other than endpoint 0(zero) support the data packets up to 1024 bytes, and support burst mode

Please refer to and call the provided subroutine library for specific applications.

# Chapter 18 Gigabit ethernet transceiver (ETH-GMAC)

The system integrates a Gigabit Ethernet Transceiver, and its Link rate can be up to 1Gbps, acting as the data link layer. In application, the hardware interface is connected to an external PHY (Gigabit/100M/speed adaptive) through RGMII or RMII signals, and it is combined with the TCP/IP protocol stack interface to implement the development of network products.

The module is designed with a MAC controller that conforms to IEEE 802.3, to cooperate with 128-bit DMA to complete the rapid transfer of MAC and SRAM data, and provide a variety of configuration options, hardware processing, status presentation and other functions.

ETH module peripheral base address: 0x4000C000

## Main features

- Compliant with IEEE 802.3 Protocol Specification and design
- Provide RGMII and RMII interfaces to connect to external Ethernet PHY
- Support 10/100/1000Mbps data transmission rate through external PHY interface
- Support full duplex operation
- Support SMI interface for configuration and management of external PHY
- The hardware automatically completes the integrity verification of IPv4 and IPv6 packets and report to the application software
- The hardware automatically completes the checksum calculation and frame length filling of IP/ICMP/UDP/TCP packets
- Several MAC address filtering modes

Please refer to and call the provided subroutine library for specific applications.

# Chapter 19 Parameters

## 19.1 Absolute maximum ratings

Stresses at or above the absolute maximum ratings listed in the table below may cause permanent damage to the device.

Table 19-1 Absolute maximum ratings

| Symbol | Parameter description | | Min. | Max. | Unit |
|---|---|---|---|---|---|
| TA | Operating ambient temperature | VDDLDO=VDDIO=3.3V | -20 | 85 | ℃ |
| | | V33USB=V33GX=3.3V | | | |
| | | V12USB=V12CORE=1.2V | | | |
| TS | Storage ambient temperature | | -55 | 125 | ℃ |
| V12CORE | System core voltage | | -0.3 | 1.5 | V |
| VDDLDO | System supply voltage | | -0.4 | 4.2 | V |
| VDDIO | GPIO port voltage | | -0.4 | 4.2 | V |
| V12USB | USB PHY signal pin voltage | | -0.3 | 1.5 | V |
| V33USB | USB PHY power voltage | | -0.4 | 4.2 | V |
| V33GX | SerDes PHY power voltage | | -0.4 | 4.2 | V |

## 19.2 Electrical characteristics

Test conditions: TA=25℃, VDDLDO=VDDIO=V33USB=V33GX=3.3V, V12CORE=V12USB=1.2V, Fsys=80MHz.

Table 19-2 Electrical characteristics

| Symbol | Parameter description | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| V12CORE | System core voltage | | 1.15 | 1.2 | 1.3 | V |
| VDDLDO | Internal LDO input power voltage | | 2.3 | 3.3 | 3.6 | V |
| VDDIO | GPIO port voltage | | 2.3 | 3.3 | 3.6 | V |
| V12USB | USB PHY low voltage supply voltage | | 1.15 | 1.2 | 1.3 | V |
| V33USB | USB PHY power voltage | | 3.15 | 3.3 | 3.5 | V |
| V33GX | SerDes PHY power voltage | | 3.0 | 3.3 | 3.6 | V |
| $I_{CC}$ | Total power current during operation, tested normally | | | 37 | | mA |
| $I_{SLP}$ | Supply current in low power mode (I/O pin output no-load or input pull-up) | Idle mode $F_{sys}$=80MHz (all peripheral clocks enabled) | | 26 | | mA |
| | | Halt mode | | 5 | | mA |
| | | Sleep Mode | | 1.3 | | mA |
| $V_{lvr}$ | Low voltage reset threshold on V33GX | | 2.6 | 2.8 | 3.0 | V |

Table 19-3 I/O characteristics

| Symbol | Parameter description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $V_{IL}$ | Input low level voltage (VDDIO=3.3V) | | | 0.8 | V |
| $V_{IH}$ | Input high level voltage (VDDIO=3.3V) | 2.0 | | | V |
| $V_{t-}$ | Schmidt trigger input negative to threshold | 0.8 | 1.1 | | V |
| $V_{t+}$ | Schmidt trigger input positive to threshold | | 1.6 | 2.0 | V |

| | | | | | |
|---|---|---|---|---|---|
| $V_{OL}$ | Output low level voltage (8/16mA input current) | | | 0.4 | V |
| $V_{OH}$ | Output high level voltage (8/16mA output current) | 2.4 | | | V |
| $R_{PU}$ | Input pull-up resistance | 40 | 75 | 190 | kΩ |
| $R_{PD}$ | Input pull-down resistance | 30 | 75 | 190 | kΩ |
| $I_{in}$ | Input leakage current | | ±1 | ±10 | uA |
| $C_{IN}$ | Input capacitance (VDDIO=2.5V or 3.3V) | 1.5 | 2.5 | 3.5 | pF |

## 19.3 Power consumption of functional modules

Test conditions: TA=25°C, VDDLDO=VDDIO=V33USB=V33GX=3.3V, V12CORE=V12USB=1.2V.

Table 19-4 Dynamic power consumption of each module

| Module \ $F_{sys}$ | 30M | 60M | 80M | 96M | 120M | Unit |
|---|---|---|---|---|---|---|
| ETH | - | - | 2 | | | mA |
| USBHS transfer | - | 28 | | | | mA |
| USBSS transfer | - | - | 110 | | | mA |
| DVP | 12 | | | | | mA |
| HSPI | 2 | 5 | 6 | 6 | 7 | mA |
| SerDes-1.2Gbps (RX and TX are enabled at the same time) | 60 | | | | | mA |
| ECDC-240MHz | 2.2 | 2.7 | 2.9 | 3.2 | 3.7 | mA |
| ECDC-160MHz | 0.6 | 1.1 | 1.4 | 1.6 | 2.2 | mA |
| TMR+UART+SPI+PWMX | 1.8 | 3.5 | 4.0 | 6.0 | 7.5 | mA |
| Core+BUS8+DMA+SRAM | 4.0 | 8.0 | 10.7 | 12.7 | 15.9 | mA |

## 19.4 Timing parameters

Test conditions: TA=25°C, VDDLDO=VDDIO=V33USB=V33GX=3.3V, V12CORE=V12USB=1.2V.

Table 19-5 Timing parameters

| Symbol | Parameter description | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $T_{rst}$ | External reset input RST# effective signal width | | 50 | $2*T_{sys}$ | - | ns |
| $T_{mr}$ | Reset delay after power on reset + load time | | 15 | 27 | 35 | ms |
| $T_{sro}$ | Reset delay after external/software reset input + load time | | 3 | 4.4 | 6 | ms |
| $T_{WAK}$ | Time used to wake up from low-power state | Idle mode | 0.3 | 0.5 | 1.5 | us |
| | | Halt mode | 20 | 27 | 40 | us |
| | | Sleep mode | | $T_{SUHSE}$+30 | | us |

Note: The values of $T_{WAK}$ in the table are based on the 80MHz system clock. If the system clock frequency is reduced, the delay is increased.

When in sleep mode, the crystal HSE start-up time needs to be considered for wake-up from low power state, so $T_{WAK}$ is increased by about 0.1-1mS (startup to available, $T_{SUHSE}$).

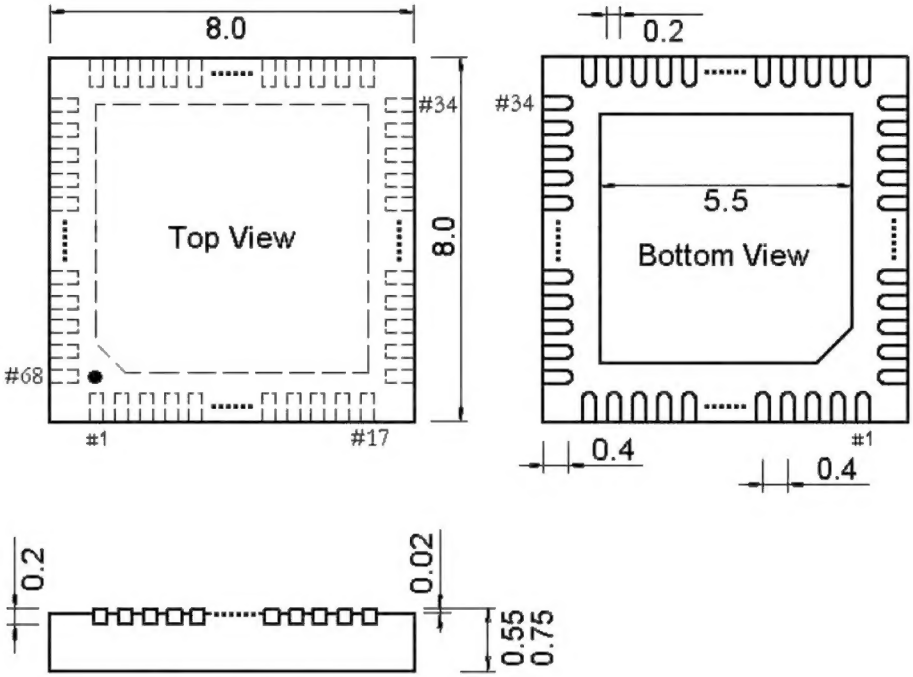
# Chapter 20 Package information

## Packages

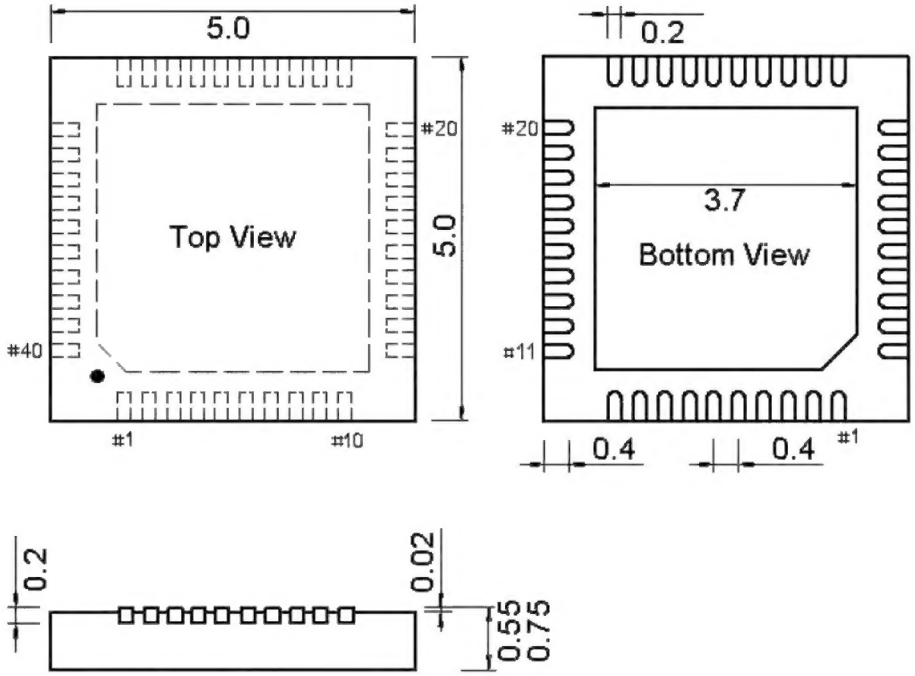
| Package | Body size | Lead pitch | | Description | Part No. |
|---|---|---|---|---|---|
| QFN68 | 8*8mm | 0.4mm | 15.75mil | Quad no-lead 68-pin | CH569W |
| QFN68 | 8*8mm | 0.4mm | 15.75mil | Quad no-lead 68-pin | CH565W |
| QFN40 | 5*5mm | 0.4mm | 15.75mil | Quad no-lead 40-pin | CH565M |

Note:
All dimensions are in millimeters.
The pin center spacing values are nominal values, with no error. The error of other dimensions is not more than ±0.5mm.

QFN68-8*8



QFN40-5*5


5.0
0.2
#20
#20
Top View
5.0
3.7
Bottom View
#40
#11
#1
#10
0.4
0.4
#1
0.2
0.02
0.55
0.75